School voor Informatietechnologie
Kennistechnologie, Informatica, Wiskunde, ICT

Uncertainty management in trajectory databases

Proefschrift voorgelegd tot het behalen van de graad van
Doctor in de Wetenschappen, richting Informatica
te verdedigen door

Walied Othman

Promotor: Prof. dr. Bart Kuijpers

19 mei 2009

Preface

This thesis would not be if not for the support of a number of people.

If I were listing the people in order of contribution, my supervisor prof. dr. Bart Kuijpers, would be leading by a wide margin. He is a supervisor like no other and went above and beyond in shaping the past four years that cumulated into this thesis.

I would also like to thank prof. dr. Harvey J. Miller, with whom I spent a six month research visit that led to many interesting results and insights, not only on a professional level.

This research has been partially funded by the tUl-impulse programme, the European Union under the FP6-IST-FET programme, Project n. FP6-14915, GeoPKDD: Geographic Privacy-Aware Knowledge Discovery and Delivery, and by the Research Foundation Flanders (FWO-Vlaanderen), Research Project G.0344.05. The latter also funded my research visit to the University of Utah and I thank them all for their support.

I spent the past four years in an office which I shared with Goele Hollanders, and I was lucky she is such a pleasant person to spend my weekdays with. I got to know Bart Moelans, Rafael Grimson and Wouter Gelade, all eager people to share their scientific enthusiasm, even beyond the realm of computer science. I also want to thank the Theoretical Computer Science Group of Hasselt University for the stimulating atmosphere they created.

Finally, my gratitude goes out to my parents and sisters for their undying support and faith in me, and to Bart Dendas, for all these years of not cutting the rope.

Diepenbeek, May 2009

Contents

1

Introduction

This thesis intersects several fields. We touch topics in Constraint Database Theory, Geographical Information Science (GIS) and even applications in Time Geography. The common denominator in these fields are moving object databases (MODs).

Nowadays more and more devices, like cell phones and GPS devices, are equipped with location aware technology (LAT). These devices, on people, vehicles or animals, produce trajectories. There are two types of trajectory data. Firstly, we have *trajectories*, which are curves in the real plane $\mathbf{R}^2$ that are parameterized by time. 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 trajectories or trajectory samples that are labelled with an identifier.

1.1 Speed and space-time prisms

The first quantity that describes an object's movement, or the lack thereof, is *speed*. Its mere definition is that speed quantifies the rate at which an object changes position over time, which is why speed and speed limits are central in this thesis.

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(see page 85 of [11]). However, linear interpolation

relies on the assumption that between sample points, a moving object moves at constant minimal speed. This assumption is realistic when sample points are frequent and occur at regular time intervals. It is more realistic to assume that moving objects have some upper bound on their speed, be it physically determined or by law such as on road networks. Given such upper bounds, an uncertainty model has been proposed which constructs *space-time prisms* between two consecutive time-space points in a trajectory sample. Basic properties of this model were discussed a few years ago in the GIS community by Pfoser and Jensen [26], Egenhofer and Hornsby [4, 5] and Miller [21], but space-time prisms were already known in the time-geography of Hägerstrand in the early 1970s [13].

If the movement of a moving object is not constrained in any direction, then a space-time prism is the intersection of two cones (one pointing upward in time and one downward in time) in time-space and all possible trajectories of the moving object between the two consecutive time-space points, given the speed bound, are located within the space-time prism. Egenhofer calls the chain of space-time prisms connecting consecutive trajectory sample points a *lifeline necklace* [4]. Figure 1.1 illustrates the concepts of space-time prism and lifeline necklace. Space-time prisms manage uncertainty more efficiently than other approaches based on cylinders, as proposed by Wolfson [33] (by a factor of three).

1.2 Complete query languages

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 relevant. Geerts proposed a model which works explicitely with the equations of motion of the moving objects, rather than with samples of trajectories, and in which the velocity of a moving object is directly available and used [9]. If we are interested in querying about speed, it is important to know which transformations of the time-space, modelled by $\mathbf{R} \times \mathbf{R}^2$, preserve the speed of a moving object. We characterise this group $\mathcal{V}$ of transformations as the combinations of affinities of time with orthogonal transformations of space composed with spatial scalings (that uses the same scale factor as the temporal affinity) and translations. Geerts et al. [10], discuss transformations that leave the velocity vector invariant, but starting from spatial transformation that are a function of time alone. Our result holds in general, for arbitrary smooth transformations of time-space. In Chapter 3, we also show that the group $\mathcal{V}$ contains precisely the transformations that preserve space-time prisms. So, the queries that involve speed are invariant

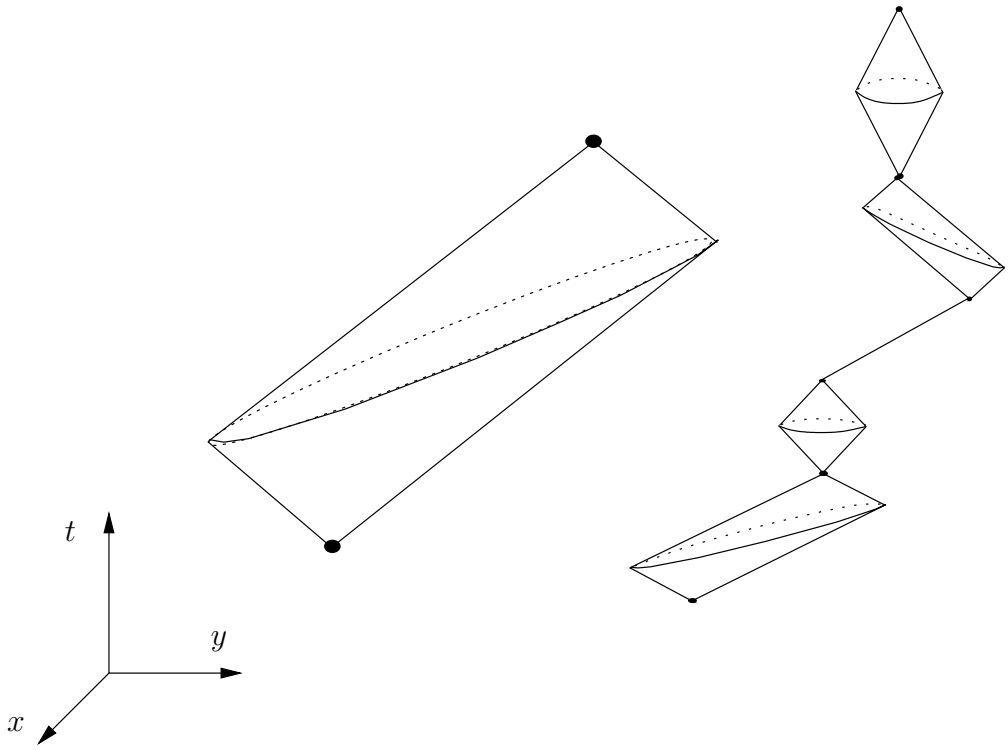


Figure 1.1: An example of a space-time prism (left) and a lifeline necklace (right).

under transformations of $\mathcal{V}$, as are queries that speak about uncertainty in term of space-time prisms. Therefore, if we are interested in querying about speed and dealing with uncertainty via space-time prisms, it is advisable to use a query language that expresses queries invariant under transformations of $\mathcal{V}$. Space-time prisms have been studied before in the context of modeling uncertainty [4, 5, 21, 26], 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 [25]. This logic is expressive enough to talk about speed and space-time prisms. 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 (modelled by positive real numbers), 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 the expression of all queries concerning speed on trajectory data and all queries concerning uncertainty in terms of space-time prisms 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 space-time prism 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 [12] and spatio-temporal databases [10]. Compared to the language proposed by Geerts et al. [10], the language we propose is far more user oriented since it is not based on geometric but speed-related predicates.

1.3 Quantifier elimination and the alibi query

A query of particular interest that has been studied by Egenhofer and Miller [4, 5, 21], is the *alibi query*. This boolean query asks whether two moving objects, that are given by samples of time-space points and speed limitations, could have physically met. This question adds up to deciding whether the lifeline necklaces of space-time prisms of these moving objects intersect or not. This problem can be considered solved in practice, when we can efficiently decide whether two space-time prisms intersect or not.

Although approximate solutions to this problem have been proposed [4], also an exact solution is possible. We show that the alibi query can be formulated in the constraint database model by means of a first-order constraint database query [17, 25]. It is well-known that first-order constraint queries can be effectively evaluated and there exist implementations of quantifier-elimination algorithms for first-order logic over the real numbers that can be used to evaluate queries, see Chapter 2 of [25]. Experiments with software packages such as QEPCAD [15], RedLog [30] and Mathematica [32] on a variety of space-time prisms show that deciding whether two concrete space-time prisms intersect can be computed on average in 2 minutes (running Windows XP Pro, SP2, with a Intel Pentium M, 1.73GHz, 1GB RAM). This means

that evaluating the alibi query on the lifeline necklaces of two moving objects that each consist of 100 space-time prisms would take, if we test intersection of space-time prisms in the two necklaces pairwise, around $100 \times 100 \times 2$ minutes, which is almost two weeks. If we would first check whether the time domains of the space-time prisms in the two necklaces overlap, we could reduce the computation time to $(100 + 100) \times 2$ minutes, or almost 7 hours. Clearly, both amounts of time are unacceptable from a practical point of view.

Another solution within the range of constraint databases is to find a formula, in which the coordinates of the apexes and limit speeds of two space-time prisms appear as *parameters*, that parametrically expresses that two space-time prisms intersect. We call this problem the *parametric alibi query*. A quantifier-free formula for this parametric version could, in theory, also be obtained by eliminating one block of three existential quantifiers using existing quantifier-elimination software packages. We have attempted this approach using MATHEMATICA and QEPCAD, but after several days of running (with the above processor), we have interrupted the computation, without successful outcome. It is known that these implementations fail on complicated, higher-dimensional problems. The benefit of having a quantifier-free first-order formula that expresses whether two space-time prisms intersect is that the alibi query on two space-time prisms can be answered in constant time. The problem of deciding whether two lifeline necklaces intersect can then be done in time proportional to the sum of the lengths of the two necklaces of space-time prisms (if we first check if the time domains of the prisms overlap).

The main contribution of Chapter 4 is the description of an analytic solution to the alibi query in isotropic two-dimensional space, thus we provide a solution to a problem that has been open since 2001 at least. We give a quantifier-free formula, that contains square roots and that expresses the (non)emptiness of the intersection of two parametrically given space-time prisms. Although, in a strict sense, this formula cannot be seen as quantifier-free first-order formula (due to the roots), it still gives the above mentioned complexity benefits. Moreover, we provide a procedure to effectively remove the roots. At the basis of our solution is a geometric theorem that describes three exclusive cases in which space-time prisms can intersect. These three cases can then be transformed into an analytic solution that can be used to answer the alibi query on the lifeline necklaces consisting of 100 space-time prisms each in less than a minute. This provides a practical solution to the alibi query.

1.4 Space-time prisms on road networks

In Chapter 5, we study moving objects and space-time prisms on *road networks*. Early adaptations of the space-time prism model to road networks were proposed by Miller [20, 22], where they introduced concepts like the *network time prism* and *potential path tree*. The first depicts all possibly visited edges and vertices in a road network, whereas the latter is a subtree of the first.

We view road networks as a graph embedding in $\mathbf{R}^2$ where all edges are embedded as straight lines between vertices. All edges have a (strictly positive) speed limit as well an associated weight, called their *time span*, which is equal to the time needed to get from one end of the edge to the other when travelling at the speed limit.

Again, a moving object is given as a finite sample of time-space points $(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i)$ where $i = 1, \ldots, N$, with $(\mathsf{x}_i, \mathsf{y}_i)$ on the road network. It is possible that also an object dependent speed limit v_i at $(\mathsf{x}_i, \mathsf{y}_i)$ is given, but we basically work with the speed limits of the road network (to cope with the former case, we simply set a uniform speed limit v_i on the road network to construct the space-time prism between sample times t_i and t_{i+1}).

The first problem we address is the computation and visualization of a space-time prism between two sample points on a road network, respecting the speed limits of the road network as well as the visualization of the spatial and temporal projection of such a space-time prism. The above mentioned time span is the key to our computation of space-time prisms on road networks. Shortest-path length in a road network is called *road network time* and is fact the shortest time to get from one point to another on the road network. If a uniform speed limit is given on the road network, road network time corresponds to shortest path distance along the road network. The algorithms that we give to compute and visualize space-time prisms and their spatial projection is based on single-source shortest path algorithms such as Dijkstra's algorithm [3]. The complexity of our algorithm is quadratic in terms of a small subset of vertices in the road network. Our algorithm for the computation of a space-time prism returns a polygonal representation in space as well as time-space. For the purpose of experiments and illustration, we implement these algorithms in MATHEMATICA [24].

Using this polygon representation of space-time prisms, we also developed an algorithm to decide the *alibi query* on road networks. This comes down to deciding whether the lifeline necklaces of two moving objects intersect or not. This problem is efficiently solved once we have an efficient way to decide whether two space-time prisms have a non-empty intersection or not. First, we develop an algorithm that decides this for moving objects moving on a straight line. In fact this algorithm is given as a first-order formula that can be applied

in constant time. Next we use our solution to the alibi query on a straight line to solve this query on a road network. Evaluating the alibi query turns out to be easy and fast (quadratic time in the number of vertices captured by both space-time prisms) once both space-time prisms have been computed. Our algorithm also computes where and when two moving objects may have met, *i.e.*, it computes the space and time projection of the intersection of the necklaces.

1.5 Space-time prisms and uncertain anchors

Up to this point space-time prisms were assumed to connect two anchors, which are time-space points. These anchors are treated as if they are exact data. The one advantage the cylinder model has over space-time prisms, is that incorporates several types of errors and not just interpolation errors. In Chapter 6 we drop the assumption that the anchors are points, and we generalise them from sample points to sample regions.

The main reason for doing so is that the sample points are never perfect measurements. While they can be highly accurate using devices that are equipped with a GPS receiver (even then the accuracy is up to a meter or more), they can also be highly inaccurate, as is the case with location data provided by cell towers. In this latter case, the uncertainty of a location can stretch out for a few miles and if we intersect this pie-shaped region (one third of a disk) with a road network, we end up with possible discontinuities in space, the error on the time-stamp of this measurement is negligible in this case. A third and final scenario, that reveals the shortcomings of a perfect data model, occurs when people are asked to retain diaries of locations and time-stamps, e.g., "I left work between 5 and 5:30 P.M." or "I was near the shopping mall around 8:30 A.M.". These cases show uncertainty about anchors both in space and time. Moreover, the latter case shows that these sample regions need not be fluently connected in time, if the shopping mall opens at 8:30 A.M., then the moving object could not be inside the mall before 8:30 A.M., but it could be after 8:30 A.M., whereas the object could be anywhere outside the mall before and after 8:30 A.M..

Sample regions model all the scenarios mentioned above. Sample regions are finite sets of box-shaped regions in space-time on top of a road network. To model the different scenarios we not only allow these regions to be not fluently connected but even disconnected, and on top of this we assume independent probability functions in time and space defined on sample regions. For example, a uniform distribution assumes all locations and time moments are equally probably, a normal distribution differentiates in the likelihood of

points.

In Chapter 6 we introduce, aside from sample regions, the envelope of an uncertain space-time prism, which contains all space-time prisms with an anchor in a starting region and one in an ending region. We also define the *emanating fraction* of a spatio-temporal point with respect to the starting regions, which is the integral of the probability functions over the subset of the starting regions from which we are able to reach this spatio-temporal point. Likewise we define the *absorbing fraction* of a spatio-temporal point with respect to the ending regions, which is the integral of the probability functions over the subset of the ending regions which we are able to reach from this spatio-temporal point. The *combined fraction* of a spatio-temporal point with respect to travel from the starting regions to the ending regions is then the product of its emanating fraction with its absorbing fraction. Furthermore, we provide algorithms to compute the structure of these concepts and visualise the envelope of an uncertain space-time prism and the emanating, absorbing and combined fraction for all time-space points inside the envelope.

2

Definitions and preliminaries

2.1 Trajectories and trajectory samples

2.1.1 Definitions and basic properties

Let $\mathbf{R}$ denote the set of the real numbers. We restrict[1] ourselves to movement in the real plane $\mathbf{R}^2$. Time-space space will be denoted by $\mathbf{R} \times \mathbf{R}^2$, where the first 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 2.1. Let $I \subseteq \mathbf{R}$ be an interval. A *trajectory* T is the graph of a piecewise-smooth[3] (with respect to t) mapping

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

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

[1] We remark that most definitions and results in this thesis 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 [4, 5, 21, 26].

[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 [23], meaning differentiable or C^1.

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 (see Chapter 3 of [11]), differentiable or even C^∞ [31]. 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 $\left\{\left(t, \frac{1-t^2}{1+t^2}, \frac{2t}{1+t^2}\right) \mid 0 \leq t \leq 1\right\}$ describes a trajectory on the quarter of the unit circle located in the first quadrant. In this example, α_x may be given by the formula $x(1+t^2) = 1 - t^2 \wedge 0 \leq t \leq 1$ and α_y may be given by the formula $x(1+t^2) = 2t \wedge 0 \leq t \leq 1$.

In practice, trajectories are only known at discrete moments in time, consider measurements from GPS-equipped devices. This partial knowledge of trajectories is formalized in the following definition. If we want to stress that some t, x, y-values (or other values) are constants, we will use sans serif characters.

Definition 2.2. A *trajectory sample* is a finite set of time-space points $\{(\mathsf{t}_0, \mathsf{x}_0, \mathsf{y}_0), (\mathsf{t}_1, \mathsf{x}_1, \mathsf{y}_1), ..., (\mathsf{t}_N, \mathsf{x}_N, \mathsf{y}_N)\}$, on which the order on time, $\mathsf{t}_0 < \mathsf{t}_1 < \cdots < \mathsf{t}_N$, induces a natural order. □

For practical purposes, we may assume that the $(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i)$-tuples of a trajectory sample contain rational values.

Definition 2.3. Let $S = \{(\mathsf{t}_0, \mathsf{x}_0, \mathsf{y}_0), (\mathsf{t}_1, \mathsf{x}_1, \mathsf{y}_1), ..., (\mathsf{t}_N, \mathsf{x}_N, \mathsf{y}_N)\}$ be a trajectory sample 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 $\mathsf{t}_0, \mathsf{t}_1, ..., \mathsf{t}_N \in I$ and $\alpha_x(\mathsf{t}_i) = \mathsf{x}_i$, $\alpha_y(\mathsf{t}_i) = \mathsf{y}_i$ for $i = 0, ..., N$. □

A classical model to reconstruct a trajectory from a sample is the *linear-interpolation model* (see Chapter 3 of [11]), 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 constructed.

Definition 2.4. For a sample $S = \{(\mathsf{t}_0, \mathsf{x}_0, \mathsf{y}_0), (\mathsf{t}_1, \mathsf{x}_1, \mathsf{y}_1), ..., (\mathsf{t}_N, \mathsf{x}_N, \mathsf{y}_N)\}$, the trajectory $LIT(S) :=$

$$\bigcup_{i=0}^{N-1} \left\{ \left(t, \frac{(\mathsf{t}_{i+1} - t)\mathsf{x}_i + (t - \mathsf{t}_i)\mathsf{x}_{i+1}}{\mathsf{t}_{i+1} - \mathsf{t}_i}, \frac{(\mathsf{t}_{i+1} - t)\mathsf{y}_i + (t - \mathsf{t}_i)\mathsf{y}_{i+1}}{\mathsf{t}_{i+1} - \mathsf{t}_i} \right) \mid \mathsf{t}_i \leq t \leq \mathsf{t}_{i+1} \right\}$$

is called the *linear-interpolation trajectory* of S. □

The functions describing the x- and y-coordinates are everywhere differentiable except possibly at the moments $\mathsf{t}_0, \mathsf{t}_1, \ldots, \mathsf{t}_N$.

2.1.2 Speed of a trajectory

Definition 2.5. 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 $\tilde{t} \in I$, then the *velocity vector of T in $\tilde{t}$* is defined as

$$\left(1, \frac{\mathrm{d}\alpha_x}{\mathrm{d}t}(\tilde{t}), \frac{\mathrm{d}\alpha_y}{\mathrm{d}t}(\tilde{t})\right)$$

and the length of the projection of this vector on the (x, y)-plane is called the *speed of T in $\tilde{t}$*. □

Let $S = \{(\mathsf{t}_0, \mathsf{x}_0, \mathsf{y}_0), (\mathsf{t}_1, \mathsf{x}_1, \mathsf{y}_1), ..., (\mathsf{t}_N, \mathsf{x}_N, \mathsf{y}_N)\}$ be a sample. Then for any t, with $\mathsf{t}_i < t < \mathsf{t}_{i+1}$, the velocity vector of $LIT(S)$ in t is $\left(1, \frac{\mathsf{x}_{i+1}-\mathsf{x}_i}{\mathsf{t}_{i+1}-\mathsf{t}_i}, \frac{\mathsf{y}_{i+1}-\mathsf{y}_i}{\mathsf{t}_{i+1}-\mathsf{t}_i}\right)$ and the corresponding speed is $\sqrt{(\mathsf{x}_{i+1}-\mathsf{x}_i)^2 + (\mathsf{y}_{i+1}-\mathsf{y}_i)^2}/(\mathsf{t}_{i+1}-\mathsf{t}_i)$, which corresponds to the minimal speed at which this distance between $(\mathsf{x}_i, \mathsf{y}_i)$ and $(\mathsf{x}_{i+1}, \mathsf{y}_{i+1})$ can be covered. At the moments $\mathsf{t}_0, \mathsf{t}_1, ..., \mathsf{t}_N$ the velocity vector and speed of $LIT(S)$ may not be defined.

2.2 A model for trajectory databases and queries

2.2.1 Trajectory and trajectory-sample databases and queries

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

Definition 2.6. 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 $(a_i, \mathsf{t}_{i,j}, \mathsf{x}_{i,j}, \mathsf{y}_{i,j})$, 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 $\{(\mathsf{t}_{i,0}, \mathsf{x}_{i,0}, \mathsf{y}_{i,0}), (\mathsf{t}_{i,1}, \mathsf{x}_{i,1}, \mathsf{y}_{i,1}), ..., (\mathsf{t}_{i,N_i}, \mathsf{x}_{i,N_i}, \mathsf{y}_{i,N_i})\}$ is a trajectory sample.

A *trajectory (sample) database* is a finite collection $\{R_1, R_2, ..., R_M\}$ of trajectory (sample) relations. □

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 2.1, we discuss how we finitely represent trajectories and trajectory samples. We will use the same representation model for trajectory relations and trajectory sample relations.

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

label	t	x	y
a	$0 \leq t \leq 1$	$1 = x(t^2+1)$	$t = y(t^2+1)$
b	$0 \leq t \leq 1$	$1 - t^2 = x(t^2+1)$	$2t = y(t^2+1)$
c	$1 \leq t \leq 2$	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	$\frac{1}{2}$	0	0
c	1	0	0
c	$\frac{3}{2}$	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 2.7. A *(sample-)trajectory database transformation* is a partial computable function from (sample-)trajectory relations to (sample-)trajectory relations. A *boolean (sample-)trajectory database query* is a partial computable function from (sample-)trajectory relations to $\{0, 1\}$. □

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

2.3 Uncertainty via space-time prisms

In 1999, Pfoser and Jensen [26] introduced the notion of *space-time prisms* in the moving object database literature to model uncertainty. Space-time prisms were later studied by Egenhofer and Hornsby [5, 4] and Miller [21]. Before Wolfson used *cylinders* to model uncertainty [11, 33].

Let S be a sample $\{(\mathsf{t}_0, \mathsf{x}_0, \mathsf{y}_0), (\mathsf{t}_1, \mathsf{x}_1, \mathsf{y}_1), \ldots, (\mathsf{t}_N, \mathsf{x}_N, \mathsf{y}_N)\}$. 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 $\{(t, x, y) \in \mathbf{R} \times \mathbf{R}^2 \mid \mathsf{t}_0 \leq t \leq \mathsf{t}_N \wedge \exists x' \exists y' (x', y') \in LIT(S) \wedge (x - x')^2 + (y - y')^2 \leq \varepsilon^2\}$.

Often, in practical applications, more is known about trajectories than merely some sample points $(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i)$. For instance, background knowledge like a physically or law imposed speed limitation v_i at location $(\mathsf{x}_i, \mathsf{y}_i)$ might be available. Such a speed limit might even depend on t_i. The speed limits that hold between two consecutive sample points can be used to model the uncertainty of a moving object's location between sample points.

More specifically, we know that at a time t, $\mathsf{t}_i \leq t \leq \mathsf{t}_{i+1}$, the object's distance to $(\mathsf{x}_i, \mathsf{y}_i)$ is at most $\mathsf{v}_i(t - \mathsf{t}_i)$ and its distance to $(\mathsf{x}_{i+1}, \mathsf{y}_{i+1})$ is at most $\mathsf{v}_i(\mathsf{t}_{i+1} - t)$. The spatial location of the object is therefore somewhere in the intersection of the disc with center $(\mathsf{x}_i, \mathsf{y}_i)$ and radius $\mathsf{v}_i(t - \mathsf{t}_i)$ and the disc with center $(\mathsf{x}_{i+1}, \mathsf{y}_{i+1})$ and radius $\mathsf{v}_i(\mathsf{t}_{i+1} - t)$. The geometric location of these points is referred to as a *space-time prism* [26, 4] and defined, for general points $(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i)$ and $(\mathsf{t}_{i+1}, \mathsf{x}_{i+1}, \mathsf{y}_{i+1})$ and speed limit v_i as follows.

Definition 2.8. Let $\mathsf{v}_i \in \mathbf{R}^+$ and $(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i), (\mathsf{t}_{i+1}, \mathsf{x}_{i+1}, \mathsf{y}_{i+1}) \in \mathbf{R} \times \mathbf{R}^2$, with $\mathsf{t}_i < \mathsf{t}_{i+1}$ and $\mathsf{v}_i \geq 0$ be given. The *space-time prism of* $(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i, \mathsf{t}_{i+1}, \mathsf{x}_{i+1}, \mathsf{y}_{i+1}, \mathsf{v}_i)$, denoted $\mathcal{P}(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i, \mathsf{t}_{i+1}, \mathsf{x}_{i+1}, \mathsf{y}_{i+1}, \mathsf{v}_i)$, is the set of points $(t, x, y) \in \mathbf{R} \times \mathbf{R}^2$ satisfying the following constraints:

$$\begin{cases} \mathsf{t}_i \leq t \leq \mathsf{t}_{i+1} \\ (x - \mathsf{x}_i)^2 + (y - \mathsf{y}_i)^2 \leq (t - \mathsf{t}_i)^2 \mathsf{v}_i^2 \\ (x - \mathsf{x}_{i+1})^2 + (y - \mathsf{y}_{i+1})^2 \leq (\mathsf{t}_{i+1} - t)^2 \mathsf{v}_i^2. \end{cases}$$

□

We call the set given by the constraints $\mathsf{t}_i \leq t$ and $(x - \mathsf{x}_i)^2 + (y - \mathsf{y}_i)^2 \leq (t - \mathsf{t}_i)^2 \mathsf{v}_i^2$ the *bottom cone* of the space-time prism and the set given by the constraints $t \leq \mathsf{t}_{i+1}$ and $(x - \mathsf{x}_{i+1})^2 + (y - \mathsf{y}_{i+1})^2 \leq (\mathsf{t}_{i+1} - t)^2 \mathsf{v}_i^2$ the *upper cone* of the space-time prism. The apices of the cones, $(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i)$ and $(\mathsf{t}_{i+1}, \mathsf{x}_{i+1}, \mathsf{y}_{i+1})$, are sometimes also referred to as anchors of the space-time prism. The axis of these cones is parallel to the t-axis. Clearly, the space-time prism is the intersection of its bottom and upper cone.

Figure 2.1 illustrates a space-time prism with $\mathsf{v}_i = 1$. For this space-time prism, the slope of the two cones is determined by the value of v_i and in this case it is 45°.

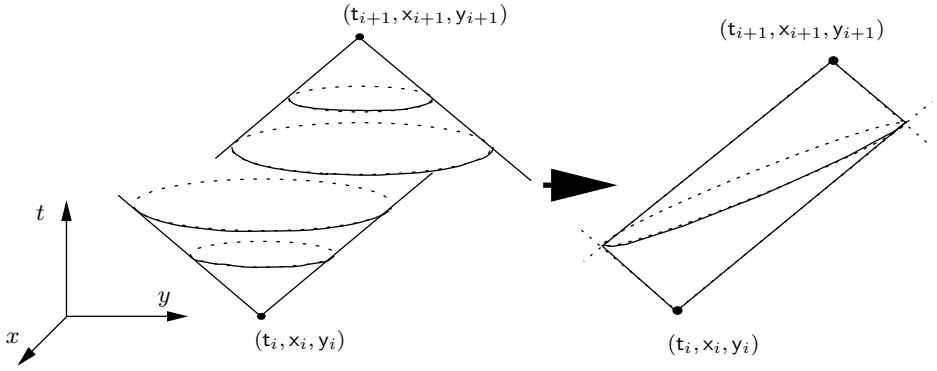


Figure 2.1: An example of a space-time prism $\mathcal{P}(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i, \mathsf{t}_{i+1}, \mathsf{x}_{i+1}, \mathsf{y}_{i+1}, \mathsf{v}_i)$.

2.3.1 Basic properties of space-time prisms

We give some basic properties of space-time prisms.

Definition 2.9. For a sample $S = \{(\mathsf{t}_0, \mathsf{x}_0, \mathsf{y}_0), (\mathsf{t}_1, \mathsf{x}_1, \mathsf{y}_1), \ldots, (\mathsf{t}_N, \mathsf{x}_N, \mathsf{y}_N)\}$ the set $\bigcup_{i=0}^{N-1} \mathcal{P}(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i, \mathsf{t}_{i+1}, \mathsf{x}_{i+1}, \mathsf{y}_{i+1}, \mathsf{v}_i)$ is called the *space-time prism chain.* □

Egenhofer calls these space-time prism chains *lifeline necklaces* [4].

The space-time prism $\mathcal{P}(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i, \mathsf{t}_{i+1}, \mathsf{x}_{i+1}, \mathsf{y}_{i+1}, \mathsf{v}_i)$ is the intersection of two cones with slope determined by v_i. At each moment t, with $\mathsf{t}_i \leq t \leq \mathsf{t}_{i+1}$, the intersection of the space-time prism 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.* At a fixed moment in time t, this disk or lens is given by the constraints

$$\begin{cases} (x - \mathsf{x}_i)^2 + (y - \mathsf{y}_i)^2 \leq (t - \mathsf{t}_i)^2 \mathsf{v}_i^2 \\ (x - \mathsf{x}_{i+1})^2 + (y - \mathsf{y}_{i+1})^2 \leq (\mathsf{t}_{i+1} - t)^2 \mathsf{v}_i^2. \end{cases}$$

The case of a lens is illustrated in Figure 2.2.

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 1. Let $(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i), (\mathsf{t}_{i+1}, \mathsf{x}_{i+1}, \mathsf{y}_{i+1})$ be time-space points with $\mathsf{t}_i < \mathsf{t}_{i+1}$ and let $\mathsf{v}_i \geq 0$. Then we have

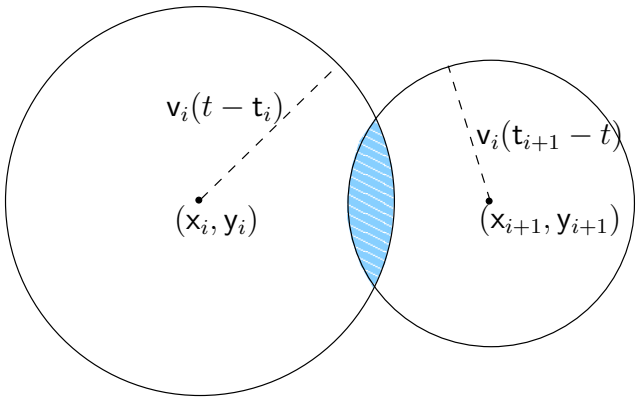


Figure 2.2: An example of a lens in a space-time prism $\mathcal{P}(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i, \mathsf{t}_{i+1}, \mathsf{x}_{i+1}, \mathsf{y}_{i+1}, \mathsf{v}_i)$ at moment t, with $\mathsf{t}_1 \leq t \leq \mathsf{t}_{i+1}$.

1. Let $d_i = \sqrt{(\mathsf{x}_i - \mathsf{x}_{i+1})^2 + (\mathsf{y}_i - \mathsf{y}_{i+1})^2}$. At any moment t between t_i and $\frac{\mathsf{t}_i+\mathsf{t}_{i+1}}{2} - \frac{d_i}{2\mathsf{v}_i}$ the space-time prism shows a disk with center $(\mathsf{x}_i, \mathsf{y}_i)$ and radius $\mathsf{v}_i(t - \mathsf{t}_i)$. At any moment between $\frac{\mathsf{t}_i+\mathsf{t}_{i+1}}{2} - \frac{d_i}{2\mathsf{v}_i}$ and $\frac{\mathsf{t}_i+\mathsf{t}_{i+1}}{2} + \frac{d_i}{2\mathsf{v}_i}$ the space-time prism is a lens and between $\frac{\mathsf{t}_i+\mathsf{t}_{i+1}}{2} + \frac{d_i}{2\mathsf{v}_i}$ and t_{i+1} it shows a disk with center $(\mathsf{x}_{i+1}, \mathsf{y}_{i+1})$ and radius $\mathsf{v}_i(\mathsf{t}_{i+1} - \mathsf{t}_i)$;

2. $d_i > \mathsf{v}_i(\mathsf{t}_{i+1} - \mathsf{t}_i)$ if and only if $\mathcal{P}(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i, \mathsf{t}_{i+1}, \mathsf{x}_{i+1}, \mathsf{y}_{i+1}, \mathsf{v}_i)$ is empty;

3. if $\mathsf{v}_i = \frac{d_i}{\mathsf{t}_{i+1}-\mathsf{t}_i}$, then $\mathcal{P}(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i, \mathsf{t}_{i+1}, \mathsf{x}_{i+1}, \mathsf{y}_{i+1}, \mathsf{v}_i)$ is a line segment in the (t, x, y)-space that is not parallel to the (x, y)-plane;

4. the space-time prism $\mathcal{P}(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i, \mathsf{t}_{i+1}, \mathsf{x}_{i+1}, \mathsf{y}_{i+1}, \mathsf{v}_i)$ is at no moment between t_i and t_{i+1} a lens (i.e., it is always a disk) if and only if $(\mathsf{x}_i, \mathsf{y}_i) = (\mathsf{x}_{i+1}, \mathsf{y}_{i+1})$ and $\mathsf{v}_i \geq 0$. In this case, $\mathcal{P}(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i, \mathsf{t}_{i+1}, \mathsf{x}_{i+1}, \mathsf{y}_{i+1}, \mathsf{v}_i)$ is the union of two cones, one with apex $(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i)$, one with apez $(\mathsf{t}_{i+1}, \mathsf{x}_i, \mathsf{y}_i)$ and both with the disk with center $(\mathsf{x}_i, \mathsf{y}_i)$ and radius $\mathsf{v}_i\left(\frac{\mathsf{t}_{i+1}+\mathsf{t}_i}{2}\right)$ at $\frac{\mathsf{t}_{i+1}+\mathsf{t}_i}{2}$ as base. □

The proof of this property is pretty straightforward and is therefor omitted. Case (4) of this property is illustrated by the space-time prism on the top right in Figure 1.1.

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

Property 2. Given time-space points $(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i), (\mathsf{t}_{i+1}, \mathsf{x}_{i+1}, \mathsf{y}_{i+1})$, with $\mathsf{t}_i < \mathsf{t}_{i+1}$ and $\mathsf{v}_i \geq 0$, the projection of the space-time prism $\mathcal{P}(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i, \mathsf{t}_{i+1}, \mathsf{x}_{i+1}, \mathsf{y}_{i+1}, \mathsf{v}_i)$ onto the (x, y)-plane is the area bordered by the ellipse with foci

$(\mathsf{x}_i, \mathsf{y}_i)$ and $(\mathsf{x}_{i+1}, \mathsf{y}_{i+1})$ and with long axis $\frac{\mathsf{v}_i(\mathsf{t}_{i+1}-\mathsf{t}_i)}{2}$. The equation of this ellipse is

$$\frac{(2x - \mathsf{x}_i - \mathsf{x}_{i+1})^2}{v^2(\mathsf{t}_{i+1} - \mathsf{t}_i)^2} + \frac{(2y - \mathsf{y}_i - \mathsf{y}_{i+1})^2}{v^2(\mathsf{t}_{i+1} - \mathsf{t}_i)^2 - (\mathsf{x}_i - \mathsf{x}_{i+1})^2 - (\mathsf{y}_i - \mathsf{y}_{i+1})^2} = 1. \quad \square$$

We denote the area bounded by the ellipse from the previous property by $\pi_{x,y}(\mathcal{P}(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i, \mathsf{t}_{i+1}, \mathsf{x}_{i+1}, \mathsf{y}_{i+1}, \mathsf{v}_i))$.

Property 3. Given $(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i), (\mathsf{t}_{i+1}, \mathsf{x}_{i+1}, \mathsf{y}_{i+1})$, with $\mathsf{t}_i < \mathsf{t}_{i+1}$ and $\mathsf{v}_i \geq 0$, then any trajectory from $(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i)$ to $(\mathsf{t}_{i+1}, \mathsf{x}_{i+1}, \mathsf{y}_{i+1})$ for which the speed at any moment $\mathsf{t}_i \leq t \leq \mathsf{t}_{i+1}$ is less than $v_{\max}$ is located within $\mathcal{P}(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i, \mathsf{t}_{i+1}, \mathsf{x}_{i+1}, \mathsf{y}_{i+1}, \mathsf{v}_i)$ and the projection of such a trajectory on the (x, y)-plane is located within $\pi_{x,y}(\mathcal{P}(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i, \mathsf{t}_{i+1}, \mathsf{x}_{i+1}, \mathsf{y}_{i+1}, \mathsf{v}_i))$. Furthermore, for any point $(\mathsf{t}, \mathsf{x}, \mathsf{y})$ in $\mathcal{P}(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i, \mathsf{t}_{i+1}, \mathsf{x}_{i+1}, \mathsf{y}_{i+1}, \mathsf{v}_i)$ there exists a trajectory from $(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i)$ to $(\mathsf{t}_{i+1}, \mathsf{x}_{i+1}, \mathsf{y}_{i+1})$ that passes through (t, x, y).

Proof. The first part of the property is trivial. For the second part, we remark that the trajectory $LIT(\{(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i), (\mathsf{t}, \mathsf{x}, \mathsf{y}), (\mathsf{t}_{i+1}, \mathsf{x}_{i+1}, \mathsf{y}_{i+1}\})$ can be taken.

Since $(\mathsf{t}, \mathsf{x}, \mathsf{y})$ is inside the space-time prism $\mathcal{P}(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i, \mathsf{t}_{i+1}, \mathsf{x}_{i+1}, \mathsf{y}_{i+1}, \mathsf{v}_i)$, which is convex, the entire line segment connecting $(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i)$ and $(\mathsf{t}, \mathsf{x}, \mathsf{y})$ is inside the prism. At time t we have that $\sqrt{(\mathsf{x} - \mathsf{x}_i)^2 + (\mathsf{y} - \mathsf{y}_i)^2} \leq \mathsf{v}_i(\mathsf{t} - \mathsf{t}_i)$, that means that a moving object's speed along the trajectory $LIT(\{(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i), (\mathsf{t}, \mathsf{x}, \mathsf{y})\})$ satisfies

$$\frac{\sqrt{(\mathsf{x} - \mathsf{x}_i)^2 + (\mathsf{y} - \mathsf{y}_i)^2}}{(\mathsf{t} - \mathsf{t}_i)} \leq \mathsf{v}_i$$

and the entire trajectory is a valid trajectory inside the space-time prism.

Likewise, the entire line segment connecting $(\mathsf{t}, \mathsf{x}, \mathsf{y})$ and $(\mathsf{t}_{i+1}, \mathsf{x}_{i+1}, \mathsf{y}_{i+1})$ is inside the prism. At time t we have that $\sqrt{(\mathsf{x}_{i+1} - \mathsf{x})^2 + (\mathsf{y}_{i+1} - \mathsf{y})^2} \leq \mathsf{v}_i(\mathsf{t}_{i+1} - \mathsf{t})$, that means that a moving object's speed along the trajectory $LIT(\{(\mathsf{t}, \mathsf{x}, \mathsf{y}), (\mathsf{t}_{i+1}, \mathsf{x}_{i+1}, \mathsf{y}_{i+1})\})$ satisfies

$$\frac{\sqrt{(\mathsf{x}_{i+1} - \mathsf{x})^2 + (\mathsf{y}_{i+1} - \mathsf{y})^2}}{(\mathsf{t}_{i+1} - \mathsf{t})} \leq \mathsf{v}_i$$

and the entire trajectory is a valid trajectory inside the space-time prism. $\square$

2.4 Road networks

Here we define road networks and trajectories on road networks.

Definition 2.10. A *road network* RN is a graph embedding in $\mathbf{R}^2$ of a labelled graph given by a finite set of vertices $\mathsf{V} = \{(x_i, y_i) \in \mathbf{R}^2 \mid i = 1, \ldots, N\}$ and a set of edges $\mathsf{E} \subseteq \mathsf{V} \times \mathsf{V}$ that are labelled by a *speed limit* and an associated *time span*. This graph embedding satisfies the following conditions. Vertices are embedded on themselves and edges are embedded as straight line segments between vertices.[4] If an edge between (x_i, y_i) and (x_j, y_j) is labeled by the speed limit $v_{ij} > 0$, then its time span w_{ij} is $\frac{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}{v_{ij}}$, i.e., it is the time needed to get from one side of an edge to another when travelling at the speed limit. □

So, we have $\mathsf{RN} =$

$$\{(x, y) = (1 - \lambda)(x_i, y_i) + \lambda(x_j, y_j) \mid \lambda \in [0, 1] \text{ and } ((x_i, y_i), (x_j, y_j)) \in \mathsf{E}\} \cup \mathsf{V}.$$

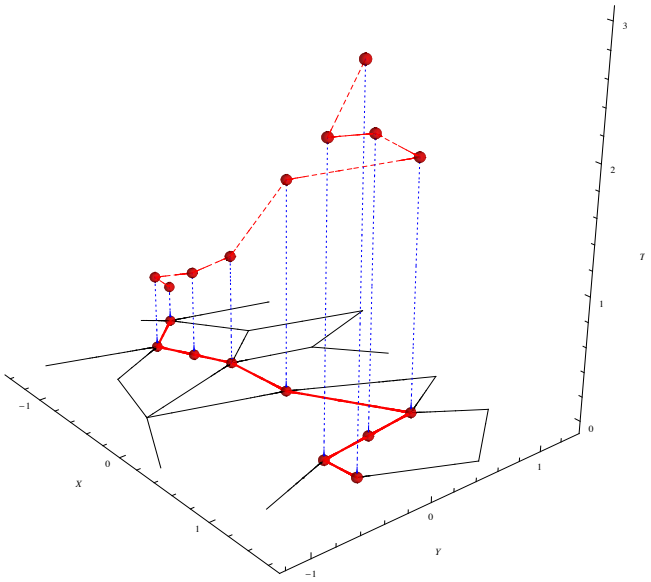


Figure 2.3: A trajectory in space-time and its projection on a road network.

A trajectory can be anything continuous, be it a polyline or something more differentiable. We assume trajectories are recorded as a discrete list of time-stamped locations, called sample points, and that nothing is known

[4]These edge embeddings may intersect. So, we can model bridges and tunnels in our model.

about an object's position between those sample points other than an upper bound on its speed.

Definition 2.11. Let T is a trajectory given by the functions α_x and α_y. T is a trajectory on a road network if it satisfies $(\alpha_x(t), \alpha_y(t)) \in \mathsf{RN}$ for all t in the time domain of T and for a trajectory sample $S = \{(t_0, x_0, y_0), (t_1, x_1, y_1), ..., (t_N, x_N, y_N)\}$ we must have $(x_i, y_i) \in \mathsf{RN}$ for all $i = 0, \ldots, N$. □

A trajectory (sample) on a road network RN is a trajectory (sample) whose spatial projection is in RN, as illustrated in Figure 2.3. Figure 2.3 shows a road network and a trajectory, as well as a sample that is consistent with the trajectory, on top of it. The spatial projections of the trajectory and the trajectory sample are also shown.

3

Complete query languages for trajectory databases

In this chapter, we focus on the trajectories that are produced by moving objects and on managing and querying them in a database. We can summarize the results of this chapter 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 remark that the completeness and soundness results presented in this chapter hold for arbitrary time-space data, but we present them for trajectory (sample) data for which the space-time prism model is specifically designed. In any case, in all the presented languages it is expressible that an output relation is a trajectory (sample) relation.

3.1 Trajectory transformations

3.1.1 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 geome-

try [23], 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 $A =_t B$. According to Allen, A is *before* B if $\mathtt{t}_A < \mathtt{t}_B$ for all $\mathtt{t}_A \in \pi_t(A)$ and all $\mathtt{t}_B \in \pi_t(B)$, 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 3.1. 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 [6]).

Property 4. 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.

Proof. This means that f preserves temporal relations between time-space points. If two time-space points p and q are co-temporal, then $f(p)$ and $f(q)$ will also be co-temporal. And if the time-space point p precedes the time-space point q in time, then $f(p)$ will also precede $f(q)$ in time. This is equivalent to the assumption that f_t is a a monotone increasing function of time alone, i.e., that $(t, x, y) \mapsto f_t(t)$ [10].

If f_t is a strictly monotone function of time alone, then f obviously preserves the order of events. And if f preserves the order of events then clearly f_t is a function of time alone, because any two co-temporal points p and q must be mapped to points that are co-temporal, that implies f_t can not be dependant on its spatial coordinates. Also, the fact that for any two time-space points p and q, $t_p < t_q$ implies that $f_t(p) < f_t(q)$ means per definition that f_t is strictly monotone increasing. □

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 5. Let T be a trajectory. If f is as above and f_t is a strictly monotone increasing function of t where $f_t'(t) > 0$ for all t, then $f(T)$ is also a trajectory.

Proof. 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 and $f_t'(t) > 0$ for all t. First, we observe that f_t satisfies all the conditions of the inverse function theorem, this guarantees that f_t^{-1} exists and is differentiable.

We have

$$f(T) = \{(f_t(t), f_x(t, \alpha_x(t), \alpha_y(t)), f_x(t, \alpha_x(t), \alpha_y(t))) \mid t \in I\}$$

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 where α_x and α_y are differentiable. □

We note that the speed vector of a trajectory in time-space always has its first component equal to one. If the condition $f_t'(t) > 0$ is omitted, then the image of a trajectory has a speed vector with its first component equal to zero in points where $f_t'(t) = 0$, in this case the image of a trajectory does not satisfy the definition of a trajectory anymore.

We remark that the restriction concerning finite representability of trajectories that we have given after Definition 2.1, 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 3.2, 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 (or tangent map of f, see Chapter 4 of [23]). 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 3.2. *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*

$$f\,(t, x, y) = \mathsf{a} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & \mathsf{a}_{11} & \mathsf{a}_{12} \\ 0 & \mathsf{a}_{21} & \mathsf{a}_{22} \end{pmatrix} \begin{pmatrix} t \\ x \\ y \end{pmatrix} + \begin{pmatrix} \mathsf{b} \\ \mathsf{b}_1 \\ \mathsf{b}_2 \end{pmatrix},$$

with $\mathsf{a}, \mathsf{b}, \mathsf{b}_1, \mathsf{b}_2 \in \mathbf{R}$, $\mathsf{a} > 0$, *and the matrix* $\begin{pmatrix} \mathsf{a}_{11} & \mathsf{a}_{12} \\ \mathsf{a}_{21} & \mathsf{a}_{22} \end{pmatrix} \in \mathbf{R}^{2\times 2}$ *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 of Theorem 3.2. 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 $\beta : \mathbf{R} \to \mathbf{R}^2 : \tau \mapsto (\beta_x(\tau), \beta_y(\tau))$, 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 5.

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 $\bar{\alpha}$ and $\bar{\beta}$ be the mappings

$$\bar{\alpha} : t \mapsto (t, \alpha_x(t), \alpha_y(t)) \text{ and } \bar{\beta} : \tau \mapsto (\tau, \beta_x(\tau), \beta_y(\tau)).$$

Since $(f \circ \bar{\alpha})(t)$ is equal to $\bar{\beta}(\tau)$, we have that the derivative[1] $(f \circ \bar{\alpha})'(t)$ 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 $f_t'(t) > 0$ 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

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

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

Since this linear transformation must be orthogonal, we have that $A \cdot A^{\mathsf{T}} = A^{\mathsf{T}} \cdot A = I$ and therefore $\det(A) = \pm 1$. These conditions lead to the following equations. Firstly,

$$\frac{\frac{\partial f_t}{\partial t}\frac{\partial f_x}{\partial t}}{\left(f_t'(t)\right)^2} = 0,$$

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

$$\frac{\left(\frac{\partial f_x}{\partial x}\right)^2 + \left(\frac{\partial f_x}{\partial y}\right)^2}{\left(f_t'(t)\right)^2} = 1$$

or

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

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 $f_t(t) = \mathsf{a}t + \mathsf{b}$ where $\mathsf{a} > 0$ since f_t is assumed to be an increasing function. The condition $\left(\frac{\partial f_x}{\partial x}\right)^2 + \left(\frac{\partial f_x}{\partial y}\right)^2 = \mathsf{a}^2$ is known as a *differential equation of light rays or the eiconal equation* [27], and has the solution $f_x(x,y) = \mathsf{a}_{11}x + \mathsf{a}_{12}y + \mathsf{b}_1$, where $\mathsf{a}_{11}^2 + \mathsf{a}_{12}^2 = \mathsf{a}^2$ and where b_1 is arbitrary. Completely analogue, we have $\left(\frac{\partial f_y}{\partial y}\right)^2 + \left(\frac{\partial f_y}{\partial x}\right)^2 = (f_t'(t))^2$ which leads to $f_y(x,y) = \mathsf{a}_{21}x + \mathsf{a}_{22}y + \mathsf{b}_2$ where $\mathsf{a}_{21}^2 + \mathsf{a}_{22}^2 = \mathsf{a}^2$ and where b_2 is arbitrary.

Thirdly,

$$\frac{\left(\frac{\partial f_x}{\partial x}\frac{\partial f_y}{\partial x} + \frac{\partial f_y}{\partial y}\frac{\partial f_x}{\partial y}\right)}{\left(f_t'(t)\right)^2} = 0.$$

And finally, $\det(A) = \pm 1$ gives $\mathsf{a}_{11}\mathsf{a}_{22} - \mathsf{a}_{12}\mathsf{a}_{21} = \pm 1$.

If we write $\mathsf{a}'_{ij} = \frac{\mathsf{a}_{ij}}{\mathsf{a}}$, then we all these equations lead to the following form of f:

$$f : \mathbf{R} \times \mathbf{R}^2 \to \mathbf{R} \times \mathbf{R}^2 : (t,x,y) \mapsto a \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & \mathsf{a}'_{11} & \mathsf{a}'_{12} \\ 0 & \mathsf{a}'_{11} & \mathsf{a}'_{22} \end{pmatrix} \begin{pmatrix} t \\ x \\ y \end{pmatrix} + \begin{pmatrix} \mathsf{b} \\ \mathsf{b}_1 \\ \mathsf{b}_2 \end{pmatrix}$$

where $\mathsf{a} > 0$, and $\begin{pmatrix} \mathsf{a}'_{11} & \mathsf{a}'_{12} \\ \mathsf{a}'_{11} & \mathsf{a}'_{22} \end{pmatrix}$ 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 [10, 6], where the same conclusion was derived, not starting from a general smooth 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 $\mathbf{R} \times \mathbf{R}^2$.

3.1.2 Transformations of space-time prisms

Suppose we transform a space-time prism $\mathcal{P}(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i, \mathsf{t}_{i+1}, \mathsf{x}_{i+1}, \mathsf{y}_{i+1}, \mathsf{v}_i)$ 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*, as before) and f_t to be strictly monotone increasing, as we have done earlier in Section 3.1.1 for trajectories. We ask ourselves which class of transformations map space-time prisms to space-time prisms. Also here we assume transformations to be smooth and bijective. Before answering this we give the following technical lemma.

Lemma 3.3. *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 $f(t) = (f(1) - f(0)) \cdot t + f(0)$.*

Proof. 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 in 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 $f(t) = \mathsf{a}t + \mathsf{b}$. Since f is assumed to be strictly monotone increasing, we must have $\mathsf{a} > 0$. Clearly, $f(0) = \mathsf{b}$ and $f(1) = \mathsf{a} + \mathsf{b} = \mathsf{a} + f(0)$. □

Theorem 3.4. *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 $(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i)$ and $(\mathsf{t}_{i+1}, \mathsf{x}_{i+1}, \mathsf{y}_{i+1})$ with $\mathsf{t}_i < \mathsf{t}_{i+1}$ and arbitrary $\mathsf{v}_i \geq 0$, $f(\mathcal{P}(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i, \mathsf{t}_{i+1}, \mathsf{x}_{i+1}, \mathsf{y}_{i+1}, \mathsf{v}_i))$ is also a space-time prism if and only if f is of the form*

$$f(t, x, y) = \begin{pmatrix} \mathsf{a} & 0 & 0 \\ 0 & \mathsf{ca}_{11} & \mathsf{ca}_{12} \\ 0 & \mathsf{ca}_{21} & \mathsf{ca}_{22} \end{pmatrix} \begin{pmatrix} t \\ x \\ y \end{pmatrix} + \begin{pmatrix} \mathsf{b} \\ \mathsf{b}_1 \\ \mathsf{b}_2 \end{pmatrix},$$

with $\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{b}_1, \mathsf{b}_2 \in \mathbf{R}$, $\mathsf{a}, \mathsf{c} > 0$, *and the matrix* $\begin{pmatrix} \mathsf{a}_{11} & \mathsf{a}_{12} \\ \mathsf{a}_{21} & \mathsf{a}_{22} \end{pmatrix} \in \mathbf{R}^{2\times 2}$ *defining an orthogonal transformation of* $\mathbf{R}^2$. *Furthermore, if these conditions are satisfied, then*

$$f(\mathcal{P}(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i, \mathsf{t}_{i+1}, \mathsf{x}_{i+1}, \mathsf{y}_{i+1}, \mathsf{v}_i)) = \mathcal{P}(f(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i), f(\mathsf{t}_{i+1}, \mathsf{x}_{i+1}, \mathsf{y}_{i+1}), \mathsf{c}\mathsf{v}_i/\mathsf{a}).$$

Proof of Theorem 3.4. Let

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

be a transformation that preserves the order of events. Suppose that for any space-time prism $\mathcal{P} = \mathcal{P}(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i, \mathsf{t}_{i+1}, \mathsf{x}_{i+1}, \mathsf{y}_{i+1}, \mathsf{v}_i)$, $f(\mathcal{P})$ is again a space-time prism.

Let us first consider the special case,

$$\mathsf{v}_i = \frac{\sqrt{(\mathsf{x}_{i+1} - \mathsf{x}_i)^2 + (\mathsf{y}_{i+1} - \mathsf{y}_i)^2}}{(\mathsf{t}_{i+1} - \mathsf{t}_i)},$$

i.e., the maximal speed is also the minimal speed. Then the space-time prism $\mathcal{P}$ is the straight line segment between $(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i)$ and $(\mathsf{t}_{i+1}, \mathsf{x}_{i+1}, \mathsf{y}_{i+1})$ in the (t, x, y)-space. This segment is not parallel to the (x, y)-plane (like all space-time prisms that are lines). Since $\mathcal{P}$ is one-dimensional and since $f(\mathcal{P})$ is assumed to be a space-time prism and since $f(\mathcal{P})$ at any moment consists of one point also $f(\mathcal{P})$ 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 space-time prism $\mathcal{P}$ with $(\mathsf{x}_i, \mathsf{y}_i) = (\mathsf{x}_{i+1}, \mathsf{y}_{i+1})$ and $\mathsf{v}_i > 0$. This space-time prism consists of a cone between t_i and $\frac{\mathsf{t}_i + \mathsf{t}_{i+1}}{2}$ with top $(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i)$ and base the disk

$$D = \left\{ \left(\frac{\mathsf{t}_i + \mathsf{t}_{i+1}}{2}, x, y \right) \mid (x - \mathsf{x}_i)^2 + (y - \mathsf{y}_i)^2 \leq \mathsf{v}_i^2 \left(\frac{\mathsf{t}_{i+1} - \mathsf{t}_i}{2} \right)^2 \right\}$$

on the one hand and a cone between $\frac{\mathsf{t}_i + \mathsf{t}_{i+1}}{2}$ and t_{i+1} with top $(\mathsf{t}_{i+1}, \mathsf{x}_i, \mathsf{y}_i)$ and the same disk D as base, on the other hand. Consider the straight line segments emanating from the top $(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i)$ and ending in some point of the central disk D. They are mapped to straight line segments in $f(\mathcal{P})$ (as we have argued before) that emanate from the top $f(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i)$ of $f(\mathcal{P})$ and that end up in some figure $f(D)$ in the hyperplane $t = f_t\left(\frac{\mathsf{t}_i + \mathsf{t}_{i+1}}{2}\right)$. Since $f(\mathcal{P})$ is assumed to be a space-time prism, the image of the bottom cone of $\mathcal{P}$ is again

a cone, and the aforementioned figure $f(D)$ in the hyperplane $t = f_t\left(\frac{\mathsf{t}_i+\mathsf{t}_{i+1}}{2}\right)$ is also a closed disk. The same holds for the top cone of $\mathcal{P}$. This half of $\mathcal{P}$ is mapped to a cone with top $f(\mathsf{t}_{i+1}, \mathsf{x}_{i+1}, \mathsf{y}_{i+1})$ and base $f(D)$. Therefore, $f(\mathcal{P})$ is the union of two cones, one with top $f(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i)$, the other with top $f(\mathsf{t}_{i+1}, \mathsf{x}_i, \mathsf{y}_i)$ and both with base $f(D)$. Since $f(\mathcal{P})$ is a space-time prism that at no moment in time is a lens, it must, by Property 1, itself be a space-time prism with equally located tops. This means that $f_x(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i) = f_x(\mathsf{t}_{i+1}, \mathsf{x}_i, \mathsf{y}_i)$ and $f_y(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i) = f_y(\mathsf{t}_{i+1}, \mathsf{x}_i, \mathsf{y}_i)$. In other words, the functions f_x and f_y are independent of t. This argument also shows that $f_t\left(\frac{\mathsf{t}_i+\mathsf{t}_{i+1}}{2}\right)$ is the middle of $f_t(\mathsf{t}_i)$ and $f_t(\mathsf{t}_{i+1})$. This means that for any t_i and t_{i+1}, $f_t\left(\frac{\mathsf{t}_i+\mathsf{t}_{i+1}}{2}\right) = \frac{1}{2}(f_t(\mathsf{t}_i) + f_t(\mathsf{t}_{i+1}))$. By Lemma 3.3, $f_t(t) = at + b$ with $a > 0$.

So, we have shown that a space-time prism-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} ($\mathsf{t}_i < \mathsf{t}_{i+1}$), the space-time prism $\mathcal{P} = \mathcal{P}(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i, \mathsf{t}_{i+1}, \mathsf{x}_i, \mathsf{y}_i, \mathsf{v}_i)$ shows a disk. Since $f(\mathcal{P})$ is again a space-time prism, 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 (u_x, u_y) and (w_x, w_y) in $\mathbf{R}^2$, $\|(u_x, u_y) - (w_x, w_y)\| = \frac{1}{\mathsf{c}}\left\|\tilde{f}(u_x, u_y) - \tilde{f}(w_x, w_y)\right\|$. Now consider $\hat{f} = \frac{1}{\mathsf{c}}\tilde{f}$, this means $\|(u_x, u_y) - (w_x, w_y)\| = \left\|\hat{f}(u_x, u_y) - \hat{f}(w_x, w_y)\right\|$ and thus $\hat{f}$ is an isometry. Just like before, cfr. Theorem 3.2, 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 plane.

If $\mathcal{P}$ is a space-time prism between the points $(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i)$ and $(\mathsf{t}_{i+1}, \mathsf{x}_{i+1}, \mathsf{y}_{i+1})$ and speed v_i, then $f(\mathcal{P}) = \mathcal{P}'$ is a space-time prism between the points $(\mathsf{t}'_i, \mathsf{x}'_i, \mathsf{y}'_i)$ and $\left(\mathsf{t}'_{i+1}, \mathsf{x}'_{i+1}, \mathsf{y}'_{i+1}\right)$ and speed $\mathsf{v}'_i = \frac{\mathsf{c}.\mathsf{v}_i}{\mathsf{a}}$, because we know that

$$\begin{cases} (x' - \mathsf{x}'_i)^2 + (y' - \mathsf{y}'_i)^2 = \mathsf{c}^2\left((x - \mathsf{x}_i)^2 + (y - \mathsf{y}_i)^2\right) \\ (t' - \mathsf{t}'_i)^2 = \mathsf{a}^2(t - t_i)^2 \end{cases}$$

has to hold for all space-time prisms, hence all v_i since degenerate space-time prisms must be transformed to degenerate space-time prisms.

This concludes the proof since it is clear that all transformations of this form also map space-time prisms to space-time prisms. □

From this results it follows that if f maps a space-time prism $\mathcal{P}$ with

maximal speed v_i to a space-time prism $f(\mathcal{P})$, the latter has maximal speed $\frac{\mathsf{cv}_i}{\mathsf{a}}$. Therefore, we can conclude the following.

Corollary 3.5. *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 space-time prisms to space-time prisms with the same speed, if and only if, f preserves the speed of trajectories (i.e., f belongs to $\mathcal{V}$ defined in Theorem 3.2).* □

Proof. Using the notation of Theorem 3.4 it follows that if f maps a space-time prism $\mathcal{P}$ with maximal speed v_i to another space-time prism and that $f(\mathcal{P})$ has maximal speed $\frac{\mathsf{cv}_i}{\mathsf{a}}$, where c is the spatial scaling factor of f and a is its temporal scaling factor. Therefore, $\mathsf{v}_i = \frac{\mathsf{cv}_i}{\mathsf{a}}$ if and only if $\mathsf{a} = \mathsf{c}$. In this case, f is a speed-preserving transformation by Theorem 3.2. □

3.2 Complete query languages for trajectory databases

3.2.1 $\mathcal{V}$-equivalent trajectory databases and $\mathcal{V}$-invariant queries

Definition 3.6. Let R and S be trajectory (sample) databases. We say that R and S are $\mathcal{V}$-*equivalent*, if there is bijection $\mu : \mathsf{Labels} \to \mathsf{Labels}$ and a speed-preserving transformation $f \in \mathcal{V}$ such that $(\mu \times f)(R) = S$. □

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

Definition 3.7. 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 $\mu : \mathsf{Labels} \to \mathsf{Labels}$ 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 $Q(R) = Q(S)$. □

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

$$\exists a \exists t_1 \exists x_1 \exists y_1 \exists t_2 \exists x_2 \exists y_2 (S(a, t_1, x_1, y_1) \wedge S(a, t_2, x_2, y_2) \wedge$$

$$(x_1 - x_2)^2 + (y_1 - y_2)^2 > 10^2(t_2 - t_1)^2)$$

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 (†)) 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 $(*)$).

3.2.2 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 3.8. 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 $x_1, ..., x_n$;
- $a = b$, where a and b are label variables; and
- $S(a, t, x, y)$ (S is 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 $\exists a$, $\forall a$, etc. □

When we instantiate the free variables in a $\mathsf{FO}(+, \times, <, 0, 1, S)$-formula $\varphi(a, b, ..., t, x, y, ...)$ by concrete values $\mathsf{a}, \mathsf{b}, ..., \mathsf{t}, \mathsf{x}, \mathsf{y}, ...$ we write $\varphi[\mathsf{a}, \mathsf{b}, ..., \mathsf{t}, \mathsf{x}, \mathsf{y}, ...]$ for the formula we obtain.

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 [28] (see also Chapter 2 of [25]). It is well known that $\mathsf{FO}(+, \times, <, 0, 1, S)$-expressible queries can be evaluated effectively (see Chapter 2 of [25]).

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 $\mathsf{FO}(+, \times, <, 0, 1, S)$-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 of Property 6. It is well known that an $\mathsf{FO}(+, \times, <, 0, 1, S)$-definable set $\{(t,x,y) \mid \varphi(t,x,y)\}$, is a semi-algebraic set (see Chapter 2 of [25]). It is expressible in $\mathsf{FO}(+, \times, <, 0, 1, S)$ 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 [8, 7]. 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 (see Chapter 2 of [25]). 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.

3.2.3 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 3.9. 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 $v_1, ..., v_n$;
- $a = b$, where a and b are label variables; 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 $\exists v$, $\forall v$, etc.

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 t_p and its spatial coordinates with respect to the standard coordinate system by x_p and y_p. The formula $S(a, p)$ expresses that a tuple (a, t_p, x_p, y_p) belongs to the input database. The atomic formula $\mathsf{Before}(p, q)$ expresses that $t_p \leq t_q$. The atomic formula $\mathsf{minSpeed}(p, q, v)$ expresses that

$$(x_p - x_q)^2 + (y_p - y_q)^2 = v^2(t_p - t_q)^2 \wedge \neg(t_q \leq t_p),$$

in other words, that v is the minimal speed to go from the spatial projection (x_p, y_p) of p to the spatial projection (x_q, y_q) of q in the time-interval $[t_p, t_q]$ 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

$$can : p \mapsto (t_p, x_p, y_p).$$

Definition 3.10. 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 $\{(a, t, x, y) \mid (id \times can)(\tilde{A}) \models \varphi(a, t, x, y)\}$.

We say that a sentence $\tilde{\varphi} \in \mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$ and a sentence $\varphi \in \mathsf{FO}(+, \times, <, 0, 1, S)$ express *equivalent* boolean queries if for any $\tilde{A}$, we have

$$\tilde{A} \models \tilde{\varphi} \text{ if and only if } (id \times can)(\tilde{A}) \models \varphi.$$

□

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 3.11. *A $\mathcal{V}$-invariant trajectory (sample) transformation or a $\mathcal{V}$-invariant 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 3.11, we introduce some more predicates on time-space points and speed values, which will come in handy later on:

- $\mathsf{inPrism}(r, p, q, v)$ expresses that $r = (r_t, r_x, r_y)$ belongs to the space-time prism $\mathcal{P}(t_p, x_p, y_p, t_q, x_q, y_q, v)$, where $p = (t_p, x_p, y_p)$ and $q = (t_q, x_q, y_q)$ (assuming that $t_p \leq t_q$);
- $\mathsf{Between}^2(p, r, q)$ 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 $\overrightarrow{p_1q_1}$ and $\overrightarrow{p_2q_2}$ 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 $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$ is $\mathsf{Between}^2$ [12]. We now show that these predicates belong to $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$.

Lemma 3.12. *The expressions listed above:* $\mathsf{inPrism}(r, p, q, v)$, $\mathsf{Between}^2(p, q, r)$, $\mathsf{Between}^{1+2}(p, q, r)$, $\mathsf{EqDist}(p_1, q_1, p_2, q_2)$, $\mathsf{Middle}(p, r, q)$ *and* $\mathsf{Perp}(p_1, q_1, p_2, q_2)$ *can all be expressed in the logic* $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$. □

Proof of Lemma 3.12. 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 $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$ 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 $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$ 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 $\mathsf{inPrism}(r, p, q, v)$, we remark that the point r lies in the space-time prism 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, $\mathsf{inPrism}(r, p, q, v)$ 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

$$=_{\mathsf{T}}(p, r) \wedge =_{\mathsf{T}}(r, q) \wedge \neg(p = r \vee r = q \vee p = q) \wedge \exists r' \exists q' \exists v (v > 0 \wedge =_{\mathsf{S}}(r, r') \wedge =_{\mathsf{S}}(q, q') \wedge \mathsf{minSpeed}(p, q', v) \wedge \mathsf{minSpeed}(p, r', v) \wedge \mathsf{minSpeed}(r', q', v) \wedge \neg\mathsf{Before}(r', p) \wedge \neg\mathsf{Before}(q', r')).$$

Indeed, the first line states that the three points p, q and r are co-temporal and distinct. Then we state that the points r' and q' have the same spatial coordinates as r and q respectively, 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 3.1. 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 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.

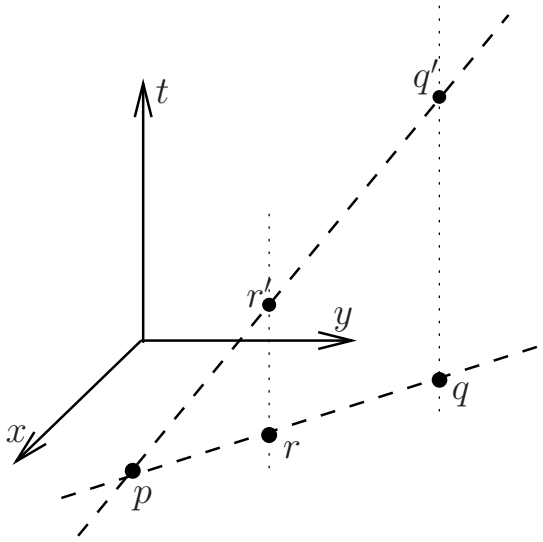


Figure 3.1: The geometric construction of $\mathsf{Between}^2$.

- The predicate $\mathsf{Between}^{1+2}$ is 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

$$\begin{aligned}\mathsf{Between}^2(p,r,q) \ \vee \ (\neg(p = r \ \vee \ r = q \ \vee \ p = q) \ \wedge \ \exists v(v > 0 \\ \mathsf{minSpeed}(p,q,v) \ \wedge \ \mathsf{minSpeed}(p,r,v) \ \wedge \ \mathsf{minSpeed}(r,q,v) \\ \wedge \ \neg\mathsf{Before}(r,p) \ \wedge \ \neg\mathsf{Before}(q,r))).\end{aligned}$$

  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

$$\begin{aligned}\exists p_1' \exists q_1' \forall r_1 \forall r_2 \exists v \ (v > 0 \ \wedge \ =_{\mathsf{T}}(p_1, q_1) \ \wedge \ =_{\mathsf{T}}(p_2, q_2) \ \wedge \\ =_{\mathsf{S}}(p_1, p_1') \ \wedge \ =_{\mathsf{T}}(p_2, p_1') \ \wedge \ =_{\mathsf{S}}(q_1, q_1') \ \wedge \ =_{\mathsf{T}}(q_2, q_1') \ \wedge \\ =_{\mathsf{T}}(r_1, r_2) \ \wedge \ =_{\mathsf{S}}(r_1, q_1') \ \wedge \ =_{\mathsf{S}}(r_2, q_2) \ \wedge \\ \neg(\mathsf{Before}(r_1, q_1') \ \vee \ \mathsf{Before}(r_2, q_2)) \\ \wedge \ \mathsf{minSpeed}(p_2, r_2, v) \ \wedge \ \mathsf{minSpeed}(p_1', r_1, v)).\end{aligned}$$

  The second and third line state 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. Then we introduce 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

$$\mathsf{Between}^2(p,r,q) \ \wedge \ \forall r' \exists v \ (v > 0 \ \wedge \ =_{\mathsf{S}}(r,r') \ \wedge \ \mathsf{Before}(r,r') \ \wedge \\ \neg =_{\mathsf{T}}(r,r') \wedge \mathsf{minSpeed}(p,r',v) \ \wedge \ \mathsf{minSpeed}(p,r',v)).$$

  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

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

  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 $\overrightarrow{p_1q_1}$ and $\overrightarrow{p_2q_2}$ are perpendicular.

□

For the purpose of the proof of Theorem 3.11, we need to generalise Definition 3.7, of $\mathcal{V}$-invariance of queries expressed by $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$-formulas.

Definition 3.13. 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 $\left(\mu^n \times f^m \times \mathrm{id}^k\right)(Q(R)) = Q(S)$. □

This definition corresponds to Definition 3.7 for transformations and boolean queries, if we take $n = m = 1$, $k = 0$ and $n = m = k = 0$, respectively.

Definition 3.14. We say that a query language is *sound* for the $\mathcal{V}$-invariant $\mathsf{FO}(+, \times, <, 0, 1, S)$-queries, if all queries are $\mathcal{V}$-invariant and expressible in $\mathsf{FO}(+, \times, <, 0, 1, S)$.

We say that a query language is *complete* for the $\mathcal{V}$-invariant $\mathsf{FO}(+, \times, <, 0, 1, S)$-queries, if all $\mathcal{V}$-invariant queries of $\mathsf{FO}(+, \times, <, 0, 1, S)$ are expressible in that language.

Now, we are ready for the proof of Theorem 3.11.

Proof of Theorem 3.11. 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 t_p, x_p and y_p 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 $S(a, t_p, x_p, y_p)$. By switching to coordinate representations, the predicates $\mathsf{minSpeed}(p, q, v)$ and $\mathsf{Before}(p, q)$ are translated to $(x_p - x_q)^2 + (y_p - y_q)^2 = v^2 (t_p - t_q)^2 \wedge t_p < t_q$ and $t_p \leq t_q$ 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 $\exists t_p \exists x_p \exists y_p$.

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 3.2 that f is the composition of a scaling by a positive factor a and an orthogonal transformation and a translation. Suppose that $f(t_p, x_p, y_p) = (p'_t, p'_x, p'_y) = p'$ and $f(t_q, x_q, y_q) = (q'_t, q'_x, q'_y) = q'$. 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 $(p'_x - q'_x)^2 + (p'_y - q'_y)^2 = a^2((x_p - x_q)^2 + (y_p - y_q)^2)$ and $v^2(p'_t - q'_t)^2 = v^2 a^2 (t_p - t_q)^2$, we have $a^2((x_p - x_q)^2 + (y_p - y_q)^2) = v^2 a^2 (t_p - t_q)^2$ or $(x_p - x_q)^2 + (y_p - y_q)^2 = v^2 (t_p - t_q)^2$. The inequality $t_p < t_q$ holds if and only if $a t_p = p'_t < a t_q = q'_t$ since $a > 0$. 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 3.13), because we apply the identity on them. 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 transformation and boolean query, expressible in $\mathsf{FO}(+, \times, <, 0, 1, S)$, can equivalently be expressed in $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$.

Since the general strategy of the proof follows the lines of Gyssens et al. [12] and Geerts et al. [6], 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 [12] for spatial data and later developed for time-space data in [10]. The details for the current setting can be easily reconstructed using the proofs in [10, 12] 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 3.12. Indeed, the formula

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

expresses that (u_0, u_1, u_2, u_3) is such a coordinate system. When traveling from u_1 to u_2 with speed 1, the elapsed time equals the elapsed space. Therefor, 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 $\mathsf{Between}^2$ [29] (see also [12]). 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 [12].

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 that states that the points t_p, x_p and y_p on u_0u_2 are the coordinates of p.

Finally, we add existential quantifiers for all the coordinate point variables and for the points u_0, u_1, u_2 and u_3. □

A corollary of Theorem 3.11 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). □

3.2.4 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 3.18, 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 [25]. We define this language here in the presence of label variables.

3.2.4.1 The language $\mathsf{FO}(+, \times, <, 0, 1, S)$+$\mathsf{while}$

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 3.15. A *program* in $\mathsf{FO}(+,\times,<,0,1,S)$+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 as a stop condition, 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-program, 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 [25]).

Theorem 3.16. *The language* $\mathsf{FO}(+,\times,<,0,1,S)$+while *is sound and complete for the computable trajectory (sample) queries (in particular, transformations or boolean queries).*

Proof of Theorem 3.16. 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 the 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. In the algorithm below, the relations T are for terms and F for formulas.

ENCODE: input=(S);
output=R_{out};

$m_S := 0$, $T := \emptyset$, $F := \emptyset$, 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
    $\forall a_1 \forall a_2 \forall a_3 \forall a_4\ (F(m_S, a_1, a_2, a_3, a_4) \leftrightarrow S(a_1, a_2, a_3, a_4))$

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 (see Chapter 2 of [25]), 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 formula that represents the output of the query.

DECODE: input=f;
output=R_{out};

$m := 0$, $T := \emptyset$, $F := \emptyset$, 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
$R_{out} := \{(r_1, r_2, r_3, r_4) \mid F(n, r_1, r_2, r_3, r_4)\}$

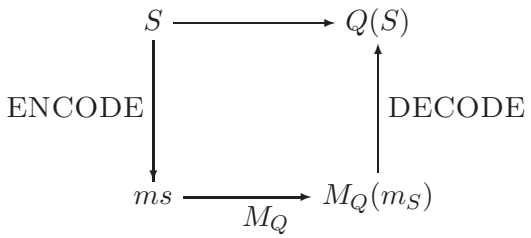
The query Q has now been effectively computed by the sequence

ENCODE;COMPUTE;DECODE;

of programs.

□

The following summarises how a query is effectively computed.



3.2.4.2 The language $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$

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 3.17. 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. □

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 2.1) 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 counter machine that computes the function, given the particular data encoding and data structures (see [25] for details).

Theorem 3.18. $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$+while *is complete for the computable $\mathcal{V}$-invariant queries on trajectory (sample) databases.* □

Proof of Theorem 3.18. We need to prove that that every computable trajectory (sample) transformation or boolean trajectory (sample) query is expressible in $\mathsf{FO}(\mathsf{Before},\mathsf{minSpeed},\tilde{S})$+while. To do this, we assume again that the label values are natural numbers. We consider, as in the proof of Theorem 3.16, an instance $\tilde{S}(a,p)$ to be given as $S\,(a,t,x,y)$, 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 $\mathsf{FO}(\mathsf{Before},\mathsf{minSpeed},\tilde{S})$+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 3.11 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 on e_0e_2 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., [12]).

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 tuples in $\tilde{T}$ are of the form $(e_0,e_1,e_2,e_3,m,p_a,t_p,x_p,y_p,w)$, where m is a time-space point on e_0e_2 denoting a natural number that encodes a formula in the variables a, t, x and y. This formula (term), when evaluated in p_a, t_p, x_p and y_p, outputs w.

The arity of $\tilde{F}$ is 9 and points in $\tilde{F}$ are of the form $(e_0,e_1,e_2,e_3,m,p_a,t_p,x_p,y_p)$, 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, t_p, x_p and y_p, gives *true*.

In all the above tuples t_p, x_p and y_p 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 proof of Theorem 3.16, 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 3.19. 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

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

□

Encoding step. We now present the encoding algorithm.

ENCODE: input=$(\tilde{S})$;
output=$(n_{Canon_{\mathcal{V}}(S)}, Type_{\mathcal{V}})$

$m := 0$, $\tilde{T} := \emptyset$, $\tilde{F} := \emptyset$, Found:=False
while ¬Found **do**
$m := m + 1$
if m encodes a **then**
$\tilde{T} := \tilde{T} \cup \{(e_0, e_1, e_2, e_3, m, p_a, t_p, x_p, y_p, p_a) \mid p_a \in e_0e_1 \text{ and } t_p, x_p, y_p \in e_0e_2\}$
else if m encodes t **then**
$\tilde{T} := \tilde{T} \cup \{(e_0, e_1, e_2, e_3, m, p_a, t_p, x_p, y_p, t_p) \mid p_a \in e_0e_1 \text{ and } t_p, x_p, y_p \in e_0e_2\}$
else if m encodes x **then**
$\tilde{T} := \tilde{T} \cup \{(e_0, e_1, e_2, e_3, m, p_a, t_p, x_p, y_p, x_p) \mid p_a \in e_0e_1 \text{ and } t_p, x_p, y_p \in e_0e_2\}$
else if m encodes y **then**
$\tilde{T} := \tilde{T} \cup \{(e_0, e_1, e_2, e_3, m, p_a, t_p, x_p, y_p, y_p) \mid p_a \in e_0e_1 \text{ and } t_p, x_p, y_p \in e_0e_2\}$
else if m encodes 0 **then**
$\tilde{T} := \tilde{T} \cup \{(e_0, e_1, e_2, e_3, m, p_a, t_p, x_p, y_p, e_0) \mid p_a \in e_0e_1 \text{ and } t_p, x_p, y_p \in e_0e_2\}$
else if m encodes 1 **then**
$\tilde{T} := \tilde{T} \cup \{(e_0, e_1, e_2, e_3, m, p_a, t_p, x_p, y_p, e_2) \mid p_a \in e_0e_1 \text{ and } t_p, x_p, y_p \in e_0e_2\}$
else if m encodes $(s + t)$ **then**
$\tilde{T} := \tilde{T} \cup \{(e_0, e_1, e_2, e_3, m, p_a, t_p, x_p, y_p, p_e) \mid \tilde{T}(e_0, e_1, e_2, e_3, enc(s), p_a, t_p, x_p, y_p, p_c) \wedge \tilde{T}(e_0, e_1, e_2, e_3, enc(t), p_a, t_p, x_p, y_p, p_d) \wedge \mathbf{Plus}(e_0, e_2, e_3, p_c, p_d, p_e)\}$
else if m encodes $(s \times t)$ **then**
$\tilde{T} := \tilde{T} \cup \{(e_0, e_1, e_2, e_3, m, p_a, t_p, x_p, y_p, p_e) \mid \tilde{T}(e_0, e_1, e_2, e_3, enc(s), p_a, t_p, x_p, y_p, p_c) \wedge \tilde{T}(e_0, e_1, e_2, e_3, enc(t), p_a, t_p, x_p, y_p, p_d) \wedge \mathbf{Times}(e_0, e_2, e_3, p_c, p_d, p_e)\}$
else if m encodes $(s \leq t)$ **then**
$\tilde{F} := \tilde{F} \cup \{(e_0, e_1, e_2, e_3, m, p_a, t_p, x_p, y_p) \mid (\exists p_c)(\exists p_d)\ (\tilde{T}(e_0, e_1, e_2, e_3, enc(s), p_a, t_p, x_p, y_p, p_c) \wedge \tilde{T}(e_0, e_1, e_2, e_3, enc(t), p_a, t_p, x_p, y_p, p_d) \wedge \mathbf{Less}(e_0, e_2, e_3, p_c, p_d)\}$
else if m encodes $(s = t)$ **then**
$\tilde{F} := \tilde{F} \cup \{(e_0, e_1, e_2, e_3, m, p_a, t_p, x_p, y_p) \mid (\exists p_c)(\exists p_d)(\tilde{T}(e_0, e_1, e_2, e_3, enc(s), p_a, t_p, x_p, y_p, p_c) \wedge \tilde{T}(e_0, e_1, e_2, e_3, enc(t), p_a, t_p, x_p, y_p, p_d) \wedge (p_c = p_d)\}$
else if m encodes $(\neg\varphi)$ **then**
$\tilde{F} := \tilde{F} \cup \{(e_0, e_1, e_2, e_3, m, p_a, t_p, x_p, y_p) \mid \neg\tilde{F}(e_0, e_1, e_2, e_3, enc(\varphi), p_a, t_p, x_p, y_p)\}$
else if m encodes $(\varphi \vee \psi)$ **then**
$\tilde{F} := \tilde{F} \cup \{(e_0, e_1, e_2, e_3, m, p_a, t_p, x_p, y_p) \mid \tilde{F}(e_0, e_1, e_2, e_3, enc(\varphi), p_a, t_p, x_p, y_p) \vee \tilde{F}(e_0, e_1, e_2, e_3, enc(\psi), p_a, t_p, x_p, y_p)\}$
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 [12], "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 m, which is the encoding of a formula, demonstrates how this can be done:

$$\begin{aligned}\forall e_0 \forall e_1 \forall e_2 \forall e_3 \exists a_{00} \exists a_{11} \exists a_{12} \exists a_{21} \exists a_{22} \exists b_0 \exists b_1 \exists b_2 (\mathsf{CoSys}(e_0, e_1, e_2, e_3) \ \wedge \\ a_{00} > 0 \ \wedge \ a_{00} a_{00} = (a_{11} a_{22} - a_{12} a_{21})(a_{11} a_{22} - a_{12} a_{21}) \ \wedge \\ \forall p_a \forall p_{t'} \forall p_{x'} \forall p_{y'} (\tilde{F}(u_0, u_1, u_2, u_3, m, p_a, p_{t'}, p_{x'}, p_{y'}) \leftrightarrow \\ \exists p \exists t_p \exists x_p \exists y_p (\tilde{S}(p_a, p) \ \wedge \ \mathsf{Coordinates}(u_0, u_1, u_2, u_3, p, t_p, x_p, y_p) \\ \wedge \ p_{t'} = a_{00} t_p + b_0 \ \wedge \ p_{x'} = a_{11} x_p + a_{12} y_p + b_1 \\ \wedge \ p_{y'} = a_{21} x_p + a_{22} y_p + b_2))).\end{aligned}$$

We note that all variables range over time-space points. For the sake of clarity, we 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

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

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 make this transformation belong to $\mathcal{V}$.

$Type_{\mathcal{V}}$ can be computed by using a similar formula but without the quantification of the variables used to parameterize the transformation and coordinate system [12], 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.

Computation step. This step can be carried out as outlined in the proof of Theorem 3.16. Since the predicate $\mathsf{Between}^2$ implicitly belongs to $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$, we can simulate all arithmetic operations on natural numbers in that language extended with a while-loop, which means we can simulate a counter program.

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 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 $F(e_0, e_1, e_2, e_3, M_Q(n_{Canon_{\mathcal{V}}(D)}), a, t_p, x_p, y_p)$.

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 [12] and sketched here:

$$\begin{aligned}\forall e_0 \forall e_1 \forall e_2 \forall e_3 \exists a_{00} \exists a_{11} \exists a_{12} \exists a_{21} \exists a_{22} \exists b_0 \exists b_1 \exists b_2\ (\mathsf{CoSys}(e_0, e_1, e_2, e_3)\ \wedge \\ \phi_{Type_{\mathcal{V}}(D)}(a_{00}, a_{11}, a_{12}, a_{21}, a_{22}, b_0, b_1, b_2)\ \wedge \\ \forall p_{t'} \forall p_{x'} \forall p_{y'}\ (\tilde{F}(e_0, e_1, e_2, e_3, M_Q(n_{Canon_{\mathcal{V}}(D)}), p_a, p_{t'}, p_{x'}, p_{y'}) \\ \wedge\ \exists t_p \exists x_p \exists y_p\ \mathsf{Coordinates}(e_0, e_1, e_2, e_3, p, t_p, x_p, y_p) \\ \wedge\ p_{t'} = a_{00}t_p + b_0\ \wedge\ p_{x'} = a_{11}x_p + a_{12}y_p + b_1 \\ \wedge\ p_{y'} = a_{21}x_p + a_{22}y_p + b_2))\end{aligned}$$

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 consecutively executing

ENCODE;COMPUTE;DECODE; .

This concludes the proof. □

3.3 Concluding remarks

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 space-time prisms 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*. We are working on understanding properties of space-time prisms on road networks. But the study of speed-preserving and space-time prism-preserving transformations on road networks and of query languages to express queries invariant under such transformations is still an unexplored field.

4

The alibi query

The *alibi query* is the boolean query which asks whether two moving objects, say with labels a and a', that are available as samples in a trajectory database, can have physically met. Since the possible positions of these moving objects are, in between sample points given speed limitations, given by space-time prisms, the alibi query asks to decide if the two space-time prism chains of a and a' intersect or not. To answer this query efficiently, it suffices to have an efficient method to decide whether two space-time prisms intersect.

In this chapter we first investigate the alibi query in dimension one, and next in dimension two.

4.1 The alibi query in dimension one

First, we present a solution to the alibi query on a straight line, i.e., on $\mathbf{R}$. This is both a warm-up to a solution to the alibi query for movement in $\mathbf{R}^2$, since some ideas that yield a solution will return there, and to a solution for the alibi query on road networks that is discussed in Chapter 5. For $\mathbf{R}$, we give a solution by means of a formula and for road networks, we describe an algorithm.

4.1.1 The alibi query for movement on a line

Let us consider a space-time prism with origin (t_o, x_o), destination (t_d, x_d) and speed limit v and one with with origin (t'_o, x'_o), destination (t'_d, x'_d) and speed

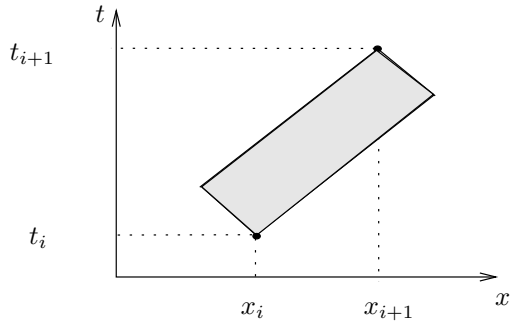


Figure 4.1: A one-dimensional space-time prism.

limit v'. The space-time prisms are then denoted by $\mathcal{P}(x_o, t_o, x_d, t_d, v)$, with $t_o \leq t_d$, and $\mathcal{P}(x'_o, t'_o, x'_d, t'_d, v')$, with $t'_o \leq t'_d$. Using the Definition 2.8 from Section 2.3, we can write the following $\mathsf{FO}(+, \times, <, 0, 1)$-formula that defines the first space-time prism:

$$\psi_{\mathcal{P}}(t, x, t_o, x_o, t_d, x_d, v) \ := \\ t_o \leq t \leq t_d \ \wedge \ (x - x_o)^2 \leq v^2(t - t_o)^2 \ \wedge \ (x - x_d)^2 \leq v^2(t - t_d)^2.$$

In this formula t_o, x_o, t_d, x_d and v appear as parameters and all tuples (t, x) that satisfy this formula belong to $\mathcal{P}(x_o, t_o, x_d, t_d, v)$. Figure 4.1 illustrates a space-time prism in one dimension.

A non-empty intersection of two space-time prisms can be satisfied in two ways:

> **Case 1**: Either one space-time prism must be entirely contained in the other; or
>
> **Case 2**: this is not the case and the borders of the two space-time prisms must intersect, which means at least one of the bordering line-segments of both space-time prisms must intersect.

Case 1: The first case, illustrated in Figure 4.2, can be translated into a formula by expressing that a top of one space-time prism is in the other, i.e.,

$$\psi_{\mathcal{P}}(t'_o, x'_o, t_o, x_o, t_d, x_d, v) \ \vee \ \psi_{\mathcal{P}}(t'_d, x'_d, t_o, x_o, t_d, x_d, v) \\ \vee \ \psi_{\mathcal{P}}(t_o, x_o, t'_o, x'_o, t'_d, x'_d, v') \ \vee \ \psi_{\mathcal{P}}(t_d, x_d, t'_o, x'_o, t'_d, x'_d, v')$$

should hold.

Case 2: The second case, illustrated in Figure 4.3, requires some more work. For this we need to compute two other points of the polygons that make up

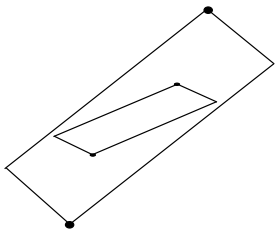


Figure 4.2: One space-time prism containing another.

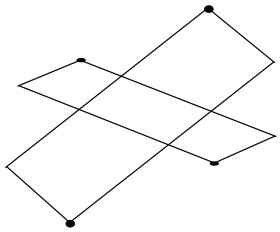


Figure 4.3: Intersecting space-time prisms.

the space-time prism, and we need to do this for each space-time prism. After that, we need to check if there exists a pair of line-segments, one from each space-time prism, that intersect.

The point (t_l, x_l), we are about to compute, is the intersection of the lines given by the following set of equations

$$\begin{cases} (x_o - x) = v\,(t - t_o) \\ (x_d - x) = v\,(t_d - t) \end{cases}$$

which is a non-singular system and therefor yields to a unique solution (given the usual assumptions that the space-time prism is non-degenerate and non-empty, cases that are treated separately, later on). We have

$$(t_l, x_l) = \left(\frac{v\,(t_d + t_o) - (x_d - x_o)}{2v}, \frac{(x_o + x_d) - v\,(t_d - t_o)}{2} \right).$$

The point (t_r, x_r) is the intersection of the lines given by the following set of equations

$$\begin{cases} (x_o - x) = -v\,(t - t_o) \\ (x_d - x) = -v\,(t_d - t)\,, \end{cases}$$

and thus,

$$(t_r, x_r) = \left(\frac{v\,(t_d + t_o) - (x_o - x_d)}{2v}, \frac{(x_o + x_d) + v\,(t_d - t_o)}{2} \right).$$

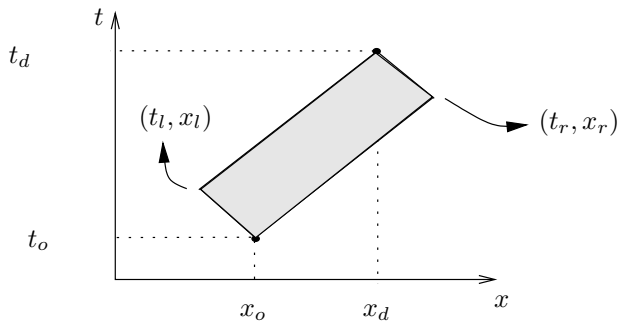


Figure 4.4: A one-dimensional space-time prism .

Similarly we get

$$\left(t'_l, x'_l\right) = \left(\frac{v'\left(t'_d + t'_o\right) - \left(x'_d - x'_o\right)}{2v'}, \frac{\left(x'_o + x'_d\right) - v'\left(t'_d - t'_o\right)}{2}\right)$$

and

$$\left(t'_r, x'_r\right) = \left(\frac{v'\left(t'_d + t'_o\right) - \left(x'_o - x'_d\right)}{2v'}, \frac{\left(x'_o + x'_d\right) + v'\left(t'_d - t'_o\right)}{2}\right).$$

These new points are visualised in Figure 4.4.

A short intermezzo. Now that we have these corner points, the only expression needed to finish the expression for the alibi query is a formula in the variables $(t_0, x_0, t_1, x_1, t'_0, x'_0, t'_1, x'_1)$ that returns true if and only if the line segment with endpoints (t_0, x_0) and (t_1, x_1) intersects the segment with endpoints (t'_0, x'_0) and (t'_1, x'_1). When we have this expression, we can write down an expression for Case 2.

So, basically we need to find an intersection, if it exists, between the lines

$$\begin{cases} (x,t) = (x_0,t_0) + \lambda\,(x_1 - x_0, t_1 - t_0) \\ (x,t) = (x'_0,t'_0) + \mu(x'_1 - x'_0, t'_1 - t'_0) \end{cases}$$

and an intersection between the segments yields to the condition $\lambda, \mu \in [0,1]$. Computing the intersection means solving the equation system

$$(x_0,t_0) + \lambda\,(x_1 - x_0, t_1 - t_0) = \left(x'_0,t'_0\right) + \mu\left(x'_1 - x'_0, t'_1 - t'_0\right)$$

which yields to the system

$$\begin{cases} (x_0 - x'_0) + \lambda\,(x_1 - x_0) = \mu\,(x'_1 - x'_0) \\ (t_0 - t'_0) + \lambda\,(t_1 - t_0) = \mu\,(t'_1 - t'_0)\,. \end{cases}$$

We want this system to have a unique solution. This is the case if it is non-singular, i.e., if and only if the determinant

$$\begin{vmatrix} x_1 - x_0 & x'_1 - x'_0 \\ t_1 - t_0 & t'_1 - t'_0 \end{vmatrix} \neq 0.$$

If this determinant is zero and the inclusion test (Case 1) failed then we can conclude the two segments do not intersect at all.

Solving for μ yields

$$\mu = \frac{(x_0 - x'_0)(t_1 - t_0) - (t_0 - t'_0)(x_1 - x_0)}{(x'_1 - x'_0)(t_1 - t_0) - (t'_1 - t'_0)(x_1 - x_0)}$$

and solving for λ yields

$$\lambda = \frac{(t_0 - t'_0)(x'_1 - x'_0) - (t'_1 - t'_0)(x_0 - x'_0)}{(t'_1 - t'_0)(x'_1 - x'_0) - (t'_1 - t'_0)(x'_1 - x'_0)}.$$

The conditions on λ and μ then read as follows: $0 \leq \mu \leq 1$ if and only if

$$0 \leq \frac{(x_0 - x'_0)(t_1 - t_0) - (t_0 - t'_0)(x_1 - x_0)}{(x'_1 - x'_0)(t_1 - t_0) - (t'_1 - t'_0)(x_1 - x_0)} \leq 1$$

and $0 \leq \lambda \leq 1$ if and only if

$$0 \leq \frac{(t_0 - t'_0)(x'_1 - x'_0) - (t'_1 - t'_0)(x_0 - x'_0)}{(t'_1 - t'_0)(x'_1 - x'_0) - (t'_1 - t'_0)(x'_1 - x'_0)} \leq 1.$$

Let $\psi_\mu(t_0, x_0, t_1, x_1, t'_0, x'_0, t'_1, x'_1)$ and $\psi_\lambda(t_0, x_0, t_1, x_1, t'_0, x'_0, t'_1, x'_1)$ be formulas expressing the above conditions.

To verify the intersection of the segment with endpoints (t_0, x_0) and (t_1, x_1) and the segment with endpoints (t'_0, x'_0) and (t'_1, x'_1), the determinant needs to be non-zero and both ψ_λ and ψ_μ need to be satisfied. Let the formula $\psi_\cap(t_0, x_0, t_1, x_1, t'_0, x'_0, t'_1, x'_1)$ express that this combination of two line segments intersects.

Note that we only need to check that a pair intersects for all pairs of opposing sides of a space-time prism. That means we need to check for every segment of the left-hand side of the first space-time prism if it intersects with a segment from the right-hand side of the second space-time prism, and vice versa. This means that Case 2 can be expressed as:

$$\begin{aligned} &\psi_\cap(t_o, x_o, t_l, x_l, t'_o, x'_o, t'_r, x'_r) \ \vee \ \psi_\cap(t_o, x_o, t_l, x_l, t'_r, x'_r, t'_d, x'_d) \\ &\quad \vee \ \psi_\cap(t_l, x_l, t_d, x_d, t'_o, x'_o, t'_r, x'_r) \ \vee \ \psi_\cap(t_l, x_l, t_d, x_d, t'_r, x'_r, t'_d, x'_d) \\ &\quad \vee \ \psi_\cap(t_o, x_o, t_r, x_r, t'_o, x'_o, t'_l, x'_l) \ \vee \ \psi_\cap(t_o, x_o, t_r, x_r, t'_l, x'_l, t'_d, x'_d) \end{aligned}$$

$$\vee\ \psi_\cap(t_r, x_r, t_d, x_d, t'_o, x'_o, t'_l, x'_l)\ \vee\ \psi_\cap(t_r, x_r, t_d, x_d, t'_l, x'_l, t'_d, x'_d).$$

Until now, we have assumed space-time prisms to be non-degenerate. If the first space-time prism is degenerate, i.e. $(x_d - x_o)^2 = v^2(t_d - t_o)^2$, and the second is not, i.e. $(x'_d - x'_o)^2 < v'^2(t'_d - t'_o)^2$, then we merely need to verify intersection for the pairs

$$\psi_\cap(t_o, x_o, t_d, x_d, t'_o, x'_o, t'_r, x'_r)\ \vee\ \psi_\cap(t_o, x_o, t_d, x_d, t'_r, x'_r, t'_d, x'_d)$$
$$\vee\ \psi_\cap(t_o, x_o, t_d, x_d, t'_o, x'_o, t'_l, x'_l)\ \vee\ \psi_\cap(t_o, x_o, t_d, x_d, t'_l, x'_l, t'_d, x'_d).$$

Finally if the both space-time prisms are degenerate, i.e. $(x_d - x_o)^2 = v^2(t_d - t_o)^2$ and $(x'_d - x'_o)^2 = v'^2(t'_d - t'_o)^2$, then we only have to test

$$\psi_\cap(t_o, x_o, t_d, x_d, t'_o, x'_o, t'_d, x'_d).$$

Combining all these cases gives an expression for the alibi query on $\mathbf{R}$. An implementation of this query in MATHEMATICA for the more general setting of movement in $\mathbf{R}^2$ can be found in [24]. The special case of movement on $\mathbf{R}$ can be obtained by placing all four points on the x-axis.

4.2 The alibi query in dimension two

We can now proceed to the alibi query in two dimensions. First we introduce an extra relation to the query language introduced in Chapter 3. This relation makes speed limits location dependent. The relation $SL(t, x, y, v)$ expresses that at (t, x, y) the reigning speed limit equals v. We will use this speed limit to constrain the object's speed between sample points. For the point based language we introduce a similar relation $\tilde{SL}(p, v)$.

We proceed to an expression for the alibi query in the $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S}, \tilde{SL})$-language. The following sections are dedicated to the intersection of two space-time prisms, where we start with a dissection of a single space-time prism, then break a possible intersection of two space-time prisms down in three cases, and give a formula that decides the alibi query.

We conclude this chapter with a formula that decides the alibi query for a fixed moment in time.

4.2.1 The alibi query

More concretely, if the trajectory with label a is given in the trajectory database by the tuples $(\mathsf{a}, \mathsf{t}_0, \mathsf{x}_0, \mathsf{y}_0), ..., (\mathsf{a}, \mathsf{t}_N, \mathsf{x}_N, \mathsf{y}_N)$ and the trajectory with label a' by the tuples $(\mathsf{a}', \mathsf{t}'_0, \mathsf{x}'_0, \mathsf{y}'_0), ..., (\mathsf{a}', \mathsf{t}'_M, \mathsf{x}'_M, \mathsf{y}'_M)$, then a has an alibi for not meeting a' if for all i, $0 \leq i \leq N-1$ and all j, $0 \leq j \leq M-1$,

$$\mathcal{P}(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i, \mathsf{t}_{i+1}, \mathsf{x}_{i+1}, \mathsf{y}_{i+1}, \mathsf{v}_i) \cap \mathcal{P}(\mathsf{t}'_j, \mathsf{x}'_j, \mathsf{y}'_j, \mathsf{t}'_{j+1}, \mathsf{x}'_{j+1}, \mathsf{y}'_{j+1}, \mathsf{v}'_j) = \emptyset. \quad (\dagger)$$

We remark that the alibi query can be expressed by a formula in the logic $\mathsf{FO}(+,\times,<,0,1,S)$, which we now give. To start, we denote the subformula

$$S(a,t_1,x_1,y_1,v_1) \ \wedge \ S(a,t_2,x_2,y_2) \ \wedge \\ \forall t_3\forall x_3\forall y_3\forall v_3 \ (S(a,t_3,x_3,y_3) \ \rightarrow \ \neg(t_1<t_3\wedge t_3<t_2)),$$

that expresses that (t_1,x_1,y_1) and (t_2,x_2,y_2) are consecutive sample points on the trajectory with label a by $\sigma(a,t_1,x_1,y_1,t_2,x_2,y_2)$.

The alibi query on a and a' is then expressed as $\varphi_{\text{alibi}}[\mathsf{a},\mathsf{a}'] =$

$$\begin{aligned}\neg \ \exists t_1\exists x_1\exists y_1\exists v_1\exists t_2\exists x_2\exists y_2\exists v_2\exists t_1'\exists x_1'\exists y_1'\exists v_1'\exists t_2'\exists x_2'\exists y_2'\exists v_2' \\ \big(\sigma(\mathsf{a},t_1,x_1,y_1,t_2,x_2,y_2) \ \wedge \ \sigma(\mathsf{a}',t_1',x_1',y_1't_2',x_2',y_2') \\ \wedge \ SL(t_1,x_1,y_1,v_1) \ \wedge \ SL(t_1',x_1',y_1',v_1') \\ \wedge \ \exists t\exists x\exists y \ (t_1\le t\le t_2 \ \wedge \ t_1'\le t\le t_2' \\ \wedge \ (x-x_1)^2+(y-y_1)^2\le (t-t_1)^2v_1^2 \\ \wedge \ (x-x_2)^2+(y-y_2)^2\le (t_2-t)^2v_1^2 \\ \wedge \ (x-x_1')^2+(y-y_1')^2\le (t-t_1')^2v_1'^2 \\ \wedge \ (x-x_2')^2+(y-y_2')^2\le (t_2'-t)^2v_1'^2)\big).\end{aligned}$$

It is well-known that $\mathsf{FO}(+,\times,<,0,1,S)$-expressible queries can be evaluated effectively on arbitrary trajectory database inputs [25, 17]. Briefly explained, this evaluation can be performed by (1) replacing the occurrences of $S(\mathsf{a},t,x,y,v)$ by a disjunction describing all the sample points belonging to the trajectory sample a; the same for a'; and (2) eliminating all the quantifiers in the obtained formula. In concreto, using the notation from above, each occurrence of $S(\mathsf{a},t,x,y,v)$ would be replaced in $\varphi_{\text{alibi}}[\mathsf{a},\mathsf{a}']$ by $\bigvee_{i=0}^{N-1}(t=\mathsf{t}_i\wedge x=\mathsf{x}_i\wedge y=\mathsf{y}_i\wedge v=\mathsf{v}_i)$, and similar for a'. This results in a (rather complicated) first-order formula over the reals $\tilde{\varphi}_{\text{alibi}}[\mathsf{a},\mathsf{a}']$ in which the predicate S does not occur any more. Since first-order logic over the reals admits the elimination of quantifiers (i.e., every formula can be equivalently expressed by a quantifier-free formula), we can decide the truth value of $\tilde{\varphi}_{\text{alibi}}[\mathsf{a},\mathsf{a}']$ by eliminating all quantifiers from this expression. In this case, we have to eliminate one block of existential quantifiers.

However, we can simplify the quantifier-elimination problem. It is easy to see, looking at (†) above, that $\neg\tilde{\varphi}_{\text{alibi}}[\mathsf{a},\mathsf{a}']$ is equivalent to

$$\bigvee_{i=0}^{N-1}\bigvee_{j=0}^{M-1}\psi_{alibi}[\mathsf{t}_i,\mathsf{x}_i,\mathsf{y}_i,\mathsf{t}_{i+1},\mathsf{x}_{i+1},\mathsf{y}_{i+1},\mathsf{v}_i,\mathsf{t}'_j,\mathsf{x}'_j,\mathsf{y}'_j,\mathsf{t}'_{j+1},\mathsf{x}'_{j+1},\mathsf{y}'_{j+1},\mathsf{v}'_j],$$

where the restricted alibi-query formula $\psi_{alibi}(t_i,x_i,y_i,t_{i+1},x_{i+1},y_{i+1},v_i,t'_j,x'_j,y'_j,t'_{j+1},x'_{j+1},y'_{j+1},v'_j)$ abbreviates the formula

$$\exists t\exists x\exists y \ (t_i\le t\le t_{i+1} \ \wedge \ t'_j\le t\le t'_{j+1} \ \wedge \ (x-x_i)^2+(y-y_i)^2\le (t-t_i)^2v_i^2$$

$$\wedge\ (x - x_{i+1})^2 + (y - y_{i+1})^2 \leq (t_{i+1} - t)^2 v_i^2$$
$$\wedge\ (x - x'_j)^2 + (y - y'_j)^2 \leq (t - t'_j)^2 v_j'^2$$
$$\wedge\ (x - x'_{j+1})^2 + (y - y'_{j+1})^2 \leq (t'_{j+1} - t)^2 v_j'^2)$$

that expresses that two space-time prisms intersect.

So, the instantiated formula

$$\psi_{alibi}[\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i, \mathsf{t}_{i+1}, \mathsf{x}_{i+1}, \mathsf{y}_{i+1}, \mathsf{v}_i, \mathsf{t}'_j, \mathsf{x}'_j, \mathsf{y}'_j, \mathsf{t}'_{j+1}, \mathsf{x}'_{j+1}, \mathsf{y}'_{j+1}, \mathsf{v}'_j]$$

expresses (†). To eliminate the existential block of quantifiers ($\exists t \exists x \exists y$) from this expression, existing software-packages for quantifier elimination, such as QEPCAD [15], REDLOG [30] and MATHEMATICA [32] can be used. We experimented QEPCAD, REDLOG and MATHEMATICA to decide if several space-time prisms intersected. The latter two programs have a similar performance and they outperform QEPCAD. To give an idea of their performance, we give some results with MATHEMATICA: the computation of $\psi_{alibi}[0, 0, 0, 1, 2, 2, \sqrt{8}, 0, 3, 3, 1, 2, 2, 2]$ takes 6 seconds; that of $\psi_{alibi}[0, 0, 0, 1, 2, 2, \sqrt{8}, 0, 3, 4, 1, 2, 2, 2]$ takes 209 seconds and the computation of $\psi_{alibi}[0, 0, 0, 1, -1, -1, 1, 0, 1, 1, 2, -1, 1, 2]$ takes 613 seconds. Roughly speaking, our experiments show that, using MATHEMATICA, this quantifier elimination can be computed on average in about 2 minutes (running Windows XP Pro, SP2, with a Intel Pentium M, 1.73GHz, 1GB RAM). This means that evaluating the alibi query on the lifeline necklaces of two moving objects that each consist of 100 space-time prisms would take around $100 \times 100 \times 2$ minutes, which is almost two weeks, when applied naively and around $(100 + 100) \times 2$ minutes or a quarter day, when first the intersection of time-intervals is tested. Clearly, in both cases, such an amount of time is unacceptable.

However, there is a better solution, which we discuss next, that can decide if two space-time prisms intersect or not in a couple of milliseconds.

4.2.2 The parametric alibi query

The uninstantiated formula

$$\psi_{alibi}(t_i, x_i, y_i, t_{i+1}, x_{i+1}, y_{i+1}, v_i, t'_j, x'_j, y'_j, t'_{j+1}, x'_{j+1}, y'_{j+1}, v'_j)$$

can be viewed as a parametric version of the restricted alibi query, where the free variables are considered parameters. This formula contains three existential quantifiers and the existing software packages for quantifier elimination could be used to obtain a quantifier-free formula $\tilde{\psi}_{alibi}(t_i, x_i, y_i, t_{i+1}, x_{i+1}, y_{i+1}, v_i, t'_j, x'_j, y'_j, t'_{j+1}, x'_{j+1}, y'_{j+1}, v'_j)$ that is equivalent to ψ_{alibi}. The formula $\tilde{\psi}_{alibi}$ could then be used to straightforwardly answer the alibi query

in time linear in its size, which is independent of the size of the input and therefore constant. We have tried to eliminate the existential block of quantifiers $\exists t \exists x \exists y$ from ψ_{alibi} using MATHEMATICA, REDLOG and QEPCAD. After some minutes of running, REDLOG invokes QEPCAD. After several days of running QEPCAD on the configuration described above, we have interrupted the computation without result. Also MATHEMATICA ran into problems without giving an answer. It is clear that eliminating a block of three existential quantifiers from a formula in 17 variables is beyond the existing quantifier-elimination implementations. Also, the instantiation of several parameters to adequately chosen constant values does not help to produce a solution. For instance, without loss of generality we can locate (t_i, x_i, y_i) in the origin $(0, 0, 0)$ and locate the other apex of the first space-time prism above the y-axis, i.e., we can take $x_{i+1} = 0$. Furthermore, we can take $v_i = 1$ and $t_{i+1} = 1$. But MATHEMATICA, REDLOG and QEPCAD cannot also not cope with this simplified situation.

The main contribution of this chapter is a the description of a quantifier-free formula equivalent to $\psi_{alibi}(t_i, x_i, y_i, t_{i+1}, x_{i+1}, y_{i+1}, v_i, t'_j, x'_j, y'_j, t'_{j+1}, x'_{j+1}, y'_{j+1}, v'_j)$. The solution we give is not a quantifier-free first-order formula in a strict sense, since it contains root expressions, but it can be easily turned into a quantifier-free first-order formula of similar length. It answers the alibi query on the lifeline necklaces of two moving objects that each consist of 100 space-time prisms in less than a minute. This description of this quantifier-free formula is the subject of Section 4.2.5.

4.2.3 $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, S)$-expression of the alibi query

In this short interlude we will show that the alibi query can be expressed in a natural way in the $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, S)$-language we introduced in Chapter 3. In that chapter, we introduced the predicate $\mathsf{inPrism}(r, p, q, v)$ that decides if a spatio-temporal point r is in the space-time prism with anchors p and q and speed limit v. The expression for $\mathsf{inPrism}(r, p, q, v)$ is

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

Using this predicate we can construct a predicate $\mathsf{prismsIntersect}(p_1, q_1, v_1, p_2, q_2, v_2)$ that decides if a space-time prism with anchors p_1 and q_1 and speed limit v_1 intersects another space-time prism with anchors p_2 and q_2 and speed limit v_2. Its expression is the following

$$\exists r \ \ (\mathsf{inPrism}(r, p_1, q_1, v_1) \ \wedge \ \mathsf{inPrism}(r, p_2, q_2, v_2)) \, .$$

One last constraint we need to build into the predicate is to ensure that the anchors of the space-time prisms we are about to test for intersection are

consecutive points in the trajectory sample. To achieve this we introduce the predicate $\mathsf{consecutive}(a, p, q)$ decides if there are no spatio-temporal points of the sample with label a between p and q. This can be expressed as

$$\forall r \, \neg \Big(\tilde{S}(a,p) \ \wedge \ \tilde{S}(a,q) \ \wedge \ \tilde{S}(a,r) \\ \wedge \ \mathsf{Before}(p,r) \ \wedge \ \mathsf{Before}(r,q) \Big) \ \vee \ (p = r \ \vee \ q = r) \, .$$

The final predicate $\mathsf{alibi}(a, b)$ takes two trajectory labels a and b as input and decides if there exists a pair of space-time prisms, one from each lifeline necklace, that intersect or not, thus answering the alibi query. Its expression is

$$\exists p_1 \exists v_{p_1} \exists q_1 \exists v_{q_1} \exists p_2 \exists v_{p_2} \exists q_2 \exists v_{q_2} \\ \big(\tilde{S}(a, p_1, v_{p_1}) \ \wedge \ \tilde{S}(a, q_1, v_{q_1}) \ \wedge \ \mathsf{consecutive}(a, p_1, q_1) \\ \wedge \ \tilde{S}(b, p_2, v_{p_2}) \ \wedge \ \tilde{S}(b, q_2, v_{q_2}) \ \wedge \ \mathsf{consecutive}(b, p_2, q_2) \\ \wedge \ \mathsf{prismsIntersect}(p_1, q_1, v_{p_1}, p_2, q_2, v_{p_2}) \big).$$

The mere length of this section shows how natural this language is to describe moving objects and their space-time prisms.

4.2.4 The geometry of space-time prisms

Before we can give an analytic solution to the alibi query and prove its correctness, we need to introduce some terminology concerning space-time prisms.

4.2.4.1 Geometric components of space-time prisms

Various geometric properties of space-time prisms [4, 17, 21] have already been described in Section 2.3. Here, we need some more definitions and notations to describe various components of a space-time prism. These components are illustrated in Figure 4.5. In this section, let $p = (t_p, x_p, y_p)$ and $q = (t_q, x_q, y_q)$ be two time-space points, with $t_p \leq t_q$ and let $v_{\max}$ be a positive real number.

The space-time prism $\mathcal{P}(p, q, v_{\max})$ is the intersection of two filled cones, given by the system of inequalities

$$\begin{cases} (x - x_p)^2 + (y - y_p)^2 \leq (t - t_p)^2 v_{\max}^2 \\ t_p \leq t \\ (x - x_q)^2 + (y - y_q)^2 \leq (t_q - t)^2 v_{\max}^2 \\ t \leq t_q \ . \end{cases}$$

The border of its *bottom cone* is the set of all points (t, x, y) that satisfy

$$\psi_{\mathsf{C}^-}(t, x, y, t_p, x_p, y_p, v_{\max}) := (x - x_p)^2 + (y - y_p)^2 = (t - t_p)^2 v_{\max}^2 \ \wedge \ t_p \leq t$$

and is denoted by $\mathsf{C}^-(p, v_{\max})$ or $\mathsf{C}^-(t_p, x_p, y_p, v_{\max})$; and the border of its *upper cone* is the set of all points (t, x, y) that satisfy

$$\psi_{\mathsf{C}^+}(t, x, y, t_q, x_q, y_q, v_{\max}) := (x - x_q)^2 + (y - y_q)^2 = (t_q - t)^2 v_{\max}^2 \ \wedge \ t \leq t_q$$

and is denoted by $\mathsf{C}^+(q, v_{\max})$ or $\mathsf{C}^+(t_q, x_q, y_q, v_{\max})$.

The set of the two apexes of $\mathcal{P}(p, q, v_{\max})$ is denoted by $\tau\mathcal{P}(p, q, v_{\max})$, i.e., $\tau\mathcal{P}(p, q, v_{\max}) = \{p, q\}$.

We call the topological border of the space-time prism $\mathcal{P}(p, q, v_{\max})$ its *mantel* and denote it by $\partial\mathcal{P}(p, q, v_{\max})$. It can be easily verified that the mantel consists of the set of points (t, x, y) that satisfy

$$\begin{aligned}
\psi_\partial(t, x, y, t_p, x_p, y_p, t_q, x_q, y_q, v_{\max}) := t_p \leq t \leq t_q \ \wedge \\
\big(2x(x_p - x_q) + x_q^2 - x_p^2 + 2y(y_p - y_q) + y_q^2 - y_p^2 \leq \\
v_{\max}^2\big(2t(t_p - t_q) + t_q^2 - t_p^2\big) \ \wedge \ (x - x_p)^2 + (y - y_p)^2 = (t - t_p)^2 v_{\max}^2\big) \\
\vee \ \big(2x(x_p - x_q) + x_q^2 - x_p^2 + 2y(y_p - y_q) + y_q^2 - y_p^2 \geq \\
v_{\max}^2\big(2t(t_p - t_q) + t_q^2 - t_p^2\big) \ \wedge \ (x - x_q)^2 + (y - y_q)^2 = (t_q - t)^2 v_{\max}^2\big).
\end{aligned}$$

The first conjunction describes the lower half of the mantel and the second conjunction describes the upper half of the mantel. The upper and lower half of the mantel are separated by a plane. The intersection of this plane with the space-time prism is an ellipse, and the border of this ellipse is what we will refer to as the *rim* of the space-time prism. We denote the rim of the space-time prism $\mathcal{P}(p, q, v_{\max})$ by $\rho\mathcal{P}(p, q, v_{\max})$ and remark that it is described by the formula

$$\begin{aligned}
\psi_\rho(t, x, y, t_p, x_p, y_p, t_q, x_q, y_q, v_{\max}) := \\
(x - x_p)^2 + (y - y_p)^2 = (t - t_p)^2 v_{\max}^2 \ \wedge \ t_p \leq t \leq t_q \\
\wedge \ 2x(x_p - x_q) + x_q^2 - x_p^2 + 2y(y_p - y_q) + y_q^2 - y_p^2 = \\
v_{\max}^2\left(2t(t_p - t_q) + t_q^2 - t_p^2\right).
\end{aligned}$$

The plane in which the rim lies splits the space-time prism into an *upper-half space-time prism* and a *bottom-half space-time prism*. The *bottom-half space-time prism* is the set of all points (t, x, y) that satisfy

$$\psi_{\mathcal{P}^-}(t,x,y,t_p,x_p,y_p,t_q,x_q,y_q,v_{\max}) :=$$
$$(x-x_p)^2+(y-y_p)^2 \le (t-t_p)^2 v_{\max}^2 \ \wedge\ t_p \le t \le t_q \ \wedge$$
$$2x(x_p-x_q)+x_q^2-x_p^2+2y(y_p-y_q)+y_q^2-y_p^2 \le$$
$$v_{\max}^2\left(2t(t_p-t_q)+t_q^2-t_p^2\right)$$

and is denoted by $\mathcal{P}^-(t_p,x_p,y_p,t_q,x_q,y_q,v_{\max})$.

The upper space-time prism is the set of all points (t,x,y) that satisfy

$$\psi_{\mathcal{P}^+}(t,x,y,t_p,x_p,y_p,t_q,x_q,y_q,v_{\max}) :=$$
$$(x-x_q)^2+(y-y_q)^2 \le (t_q-t)^2 v_{\max}^2 \ \wedge\ t_p \le t \le t_q \ \wedge$$
$$2x(x_p-x_q)+x_q^2-x_p^2+2y(y_p-y_q)+y_q^2-y_p^2 \ge$$
$$v_{\max}^2\left(2t(t_p-t_q)+t_q^2-t_p^2\right)$$

and is denoted by $\mathcal{P}^+(t_p,x_p,y_p,t_q,x_q,y_q,v_{\max})$.

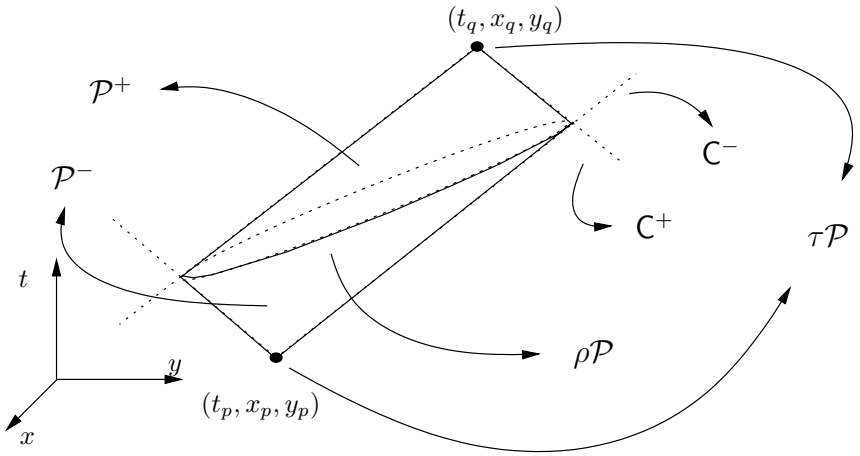


Figure 4.5: The components of the space-time prism $\mathcal{P}(t_p,x_p,y_p,t_q,x_q,y_q,v_{\max})$.

4.2.4.2 The intersection of two cones

Let $\mathsf{C}^-(t_1,x_1,y_1,v_1)$ and $\mathsf{C}^-(t_2,x_2,y_2,v_2)$ be two bottom cones. A bottom cone, e.g., $\mathsf{C}^-(t_1,x_1,y_1,v_1)$, can be seen as a circle in 2-dimensional space (x,y)-space with center (x_1,y_1) and linearly growing radius $(t-t_1)v_1$ as $t_1 \le t$.

Let us assume that the apex of neither of these cones is inside the other cone, i.e.,

$$\big((x_1-x_2)^2+(y_1-y_2)^2 > (t_1-t_2)^2 v_1^2 \ \vee \ t_1 < t_2\big) \\ \wedge \ \big((x_1-x_2)^2+(y_1-y_2)^2 > (t_1-t_2)^2 v_2^2 \ \vee \ t_2 < t_1\big)$$

is satisfied.

This assumption implies that at t_1 and t_2 neither radius is larger than or equal to the distance between the two cone centers. So, at first the two circles are disjoint and after growing for some time they intersect in one point. We call the first (in time) time-space point where the two circles touch in a single point, and thus for which the sum of the two radii is equal to the distance between the two centers the *initial contact* of the two cones $\mathsf{C}^-(t_1, x_1, y_1, v_1)$ and $\mathsf{C}^-(t_2, x_2, y_2, v_2)$. It is the unique point (t, x, y) that satisfies the formula

$$\begin{aligned}\psi_{IC^-}(t,x,y,t_1,x_1,y_1,v_1,t_2,x_2,y_2,v_2) := {} & t_1 \le t \ \wedge \ t_2 \le t \ \wedge \\ & (x-x_1)^2+(y-y_1)^2 = (t-t_1)^2 v_1^2 \ \wedge \\ & (x-x_2)^2+(y-y_2)^2 = (t-t_2)^2 v_2^2 \ \wedge \\ & ((t-t_1)v_1+(t-t_2)v_2)^2 = (x_1-x_2)^2+(y_1-y_2)^2.\end{aligned}$$

The initial contact of two cones $\mathsf{C}^+(t_1, x_1, y_1, v_1)$ and $\mathsf{C}^+(t_2, x_2, y_2, v_2)$ is given by the formula $\psi_{IC^+}(t, x, y, t_1, x_1, y_1, v_1, t_2, x_2, y_2, v_2)$ that we obtain from ψ_{IC^-} by replacing $(t_1 \le t \ \wedge \ t_2 \le t)$ with $(t \le t_1 \ \wedge \ t \le t_2)$.

$$\begin{aligned}\psi_{IC^+}(t,x,y,t_1,x_1,y_1,v_1,t_2,x_2,y_2,v_2) := {} & t \le t_1 \ \wedge \ t \le t_2 \ \wedge \\ & (x-x_1)^2+(y-y_1)^2 = (t-t_1)^2 v_1^2 \ \wedge \\ & (x-x_2)^2+(y-y_2)^2 = (t-t_2)^2 v_2^2 \ \wedge \\ & ((t-t_1)v_1+(t-t_2)v_2)^2 = (x_1-x_2)^2+(y_1-y_2)^2.\end{aligned}$$

We denote the singleton sets containing the initial contacts by $\mathsf{IC}(\mathsf{C}^-(t_1, x_1, y_1, v_1), \mathsf{C}^-(t_2, x_2, y_2, v_2))$ and $\mathsf{IC}(\mathsf{C}^+(t_1, x_1, y_1, v_1), \mathsf{C}^+(t_2, x_2, y_2, v_2))$.

From the last equation in of the system in ψ_{IC^-} and ψ_{IC^+}, we easily obtain

$$t = \frac{\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}+t_1v_1+t_2v_2}{v_1+v_2}.$$

To compute the other two coordinates (x, y) of the initial contact, we observe that, in the plane $t = (\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}+t_1v_1+t_2v_2)/(v_1+v_2)$, it lies on the line segment bounded by (x_1, y_1) and (x_2, y_2) and that its distance from (x_1, y_1) equals $v_1(t-t_1)$ and its distance from (x_1, y_1) equals $v_2(t-t_2)$.

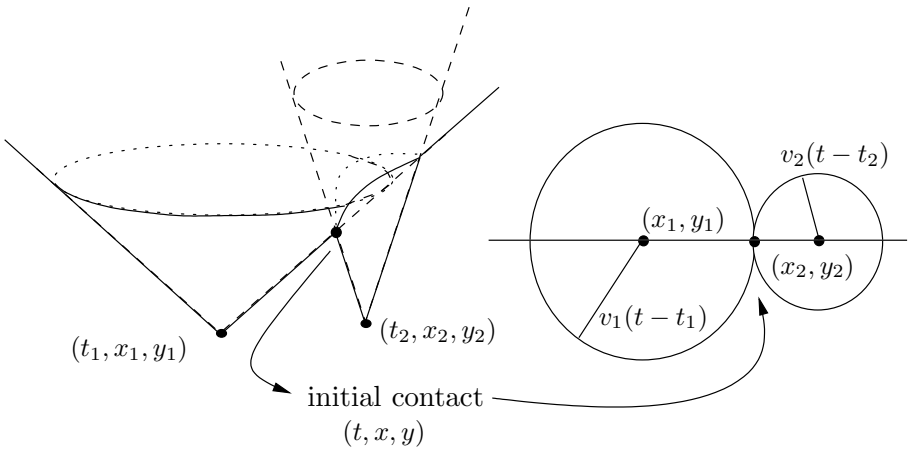


Figure 4.6: Intersecting cones and their initial contact (3-dimensional view on the left and 2-dimensional view from above on the right).

We can conclude that the initial contact has (t, x, y)-coordinates given by the following system of equations

$$\begin{cases} t &= \frac{\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}+t_1v_1+t_2v_2}{v_1+v_2} \\ x &= x_1 + v_1(t-t_1)\frac{x_2-x_1}{\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}} \\ y &= y_1 + v_1(t-t_1)\frac{y_2-y_1}{\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}}. \end{cases}$$

The point in the singleton set $\mathsf{IC}(\mathsf{C}^+(t_1, x_1, y_1, v_1), \mathsf{C}^+(t_2, x_2, y_2, v_2))$ has similar coordinates,

$$\begin{cases} t &= \frac{\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}+t_1v_1+t_2v_2}{v_1+v_2} \\ x &= x_1 + v_1(t_1-t)\frac{x_2-x_1}{\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}} \\ y &= y_1 + v_1(t_1-t)\frac{y_2-y_1}{\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}}. \end{cases}$$

This means that we can give more explicit descriptions to replace ψ_{IC^-} and ψ_{IC^+}.

4.2.5 An analytic solution to the alibi query

In this section, we first describe the solution to the alibi query on a geometric level. Next, we prove its correctness and transform it into an analytic solution and finally we show how to construct a quantifier-free first-order formula from the analytic solution.

4.2.5.1 Preliminary geometric considerations

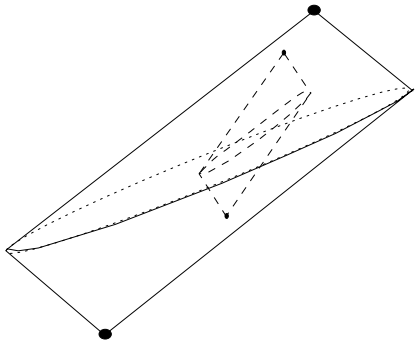


Figure 4.7: One space-time prism is contained in the other.

The solution we present is based on the observation that the two main cases of intersection (that do not exclude each other) are: (1) an apex of one space-time prism is in the other; and (2) the mantels of the space-time prisms intersect. This follows from the fact that intersection either means containment or a proper intersection where the borders of the sets intersect.

The inclusion of one space-time prism in the other, illustrated in Figure 5.4, is an example of the first case. It is clear that if no apex is contained in another space-time prism and we still assume that the space-time prisms intersect, than their mantels must intersect. We show this more formally in Lemma 4.1. In this second case, the idea is to find a special point (a witness point) that is easily computable and necessarily in the intersection.

Let us consider two space-time prisms $\mathcal{P}(t_1, x_1, y_1, t_1', x_1', y_1', v_1)$ and $\mathcal{P}(t_2, x_2, y_2, t_2', x_2', y_2', v_2)$ with bottom cones $\mathsf{C}^-(t_1, x_1, y_1, v_1)$ and $\mathsf{C}^-(t_2, x_2, y_2, v_2)$ and let us assume that none of the apices is inside the other cone. One special point is the point of initial contact $\mathsf{IC}(\mathsf{C}^-(t_1, x_1, y_1, v_1), \mathsf{C}^-(t_2, x_2, y_2, v_2))$. However, this point can not be guaranteed to be in the intersection if the mantels of the two space-time prisms intersect, as we will show in the following example. Consider two space-time prisms $\mathcal{P}(0, 0, 0, 4, 0, 0, 1)$ and $\mathcal{P}(0, 2, 0, 4, 2, 0, 1)$ with bottom cones $\mathsf{C}^-(0, 0, 0, 1)$ and $\mathsf{C}^-(0, 2, 0, 1)$. The intersection is a hyperbola in the plane $x = 1$ with equation $t^2 - y^2 = 1$. The initial contact of the two bottom cones is the point $(1, 0, 1)$. To show that this point of initial contact does not need to be in the intersection of the two space-time prisms, the idea is to cut this point out of the intersection as follows. Suppose one space-time prism has apexes, $(0, 0, 0)$ and (a, b, c) and speed 1. The plane in which its rim lies is given by $-2ax + a^2 - 2by + b^2 + 2ct - c^2 = 0$. This plane cuts the plane α given by the equality $x = 1$ in a line given by the equation $-2by + 2ct - 2a + a^2 - c^2 = 0$. Clearly, we can choose (a, b, c) such that the

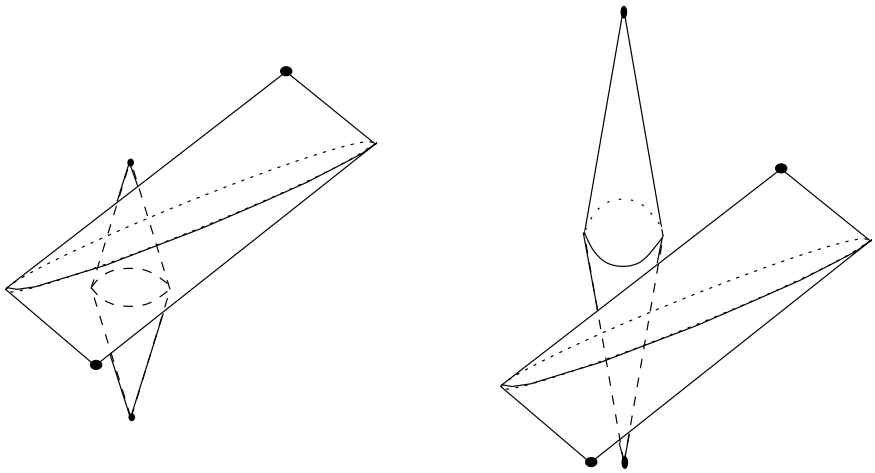


Figure 4.8: Clean cut between cones.

line contains the points $\left(\frac{\sqrt{5}}{2}, 1, \frac{1}{2}\right)$ and $\left(\sqrt{2}, 1, 1\right)$. Everything below this line will be part of the first space-time prism and the second cone, but the initial contact is situated above the line, effectively cutting it out of the intersection. All this is illustrated in Figure 4.9.

We notice how the plane in which the rim lies and the rim itself is the evil do-er. If neither rim intersects the mantel of the other space-time prism, then the intersection of mantels is the same as an intersection of cones. In which case the initial contact will not be cut out and can be used to determine if there is intersection in this manner.

Using contraposition on the statement in the previous paragraph we get: *if there is an intersection and no initial contact is in the intersection then a rim must intersect the other space-time prism's mantel.*

To verify intersection with the apexes and initial contacts is straightforward. Verifying if a rim intersects a mantel results in solving a quartic polynomial equation in one variable and verifying the solution in a single inequality in which no variable appears with a degree higher than one.

4.2.5.2 Outline of the solution

Suppose, for the remainder of this section, we wish to verify if the space-time prisms $\mathcal{P}_1 = \mathcal{P}(t_1, x_1, y_1, t_2, x_2, y_2, v_1)$ and $\mathcal{P}_2 = \mathcal{P}(t_3, x_3, y_3, t_4, x_4, y_4, v_2)$ intersect. Moreover, we assume the space-time prisms are non-empty, i.e., the system

$$\begin{cases} (x_2 - x_1)^2 + (y_2 - y_1)^2 \ \leq \ (t_2 - t_1)^2 v_1^2 \\ (x_4 - x_3)^2 + (y_4 - y_3)^2 \ \leq \ (t_4 - t_3)^2 v_2^2 \end{cases}$$

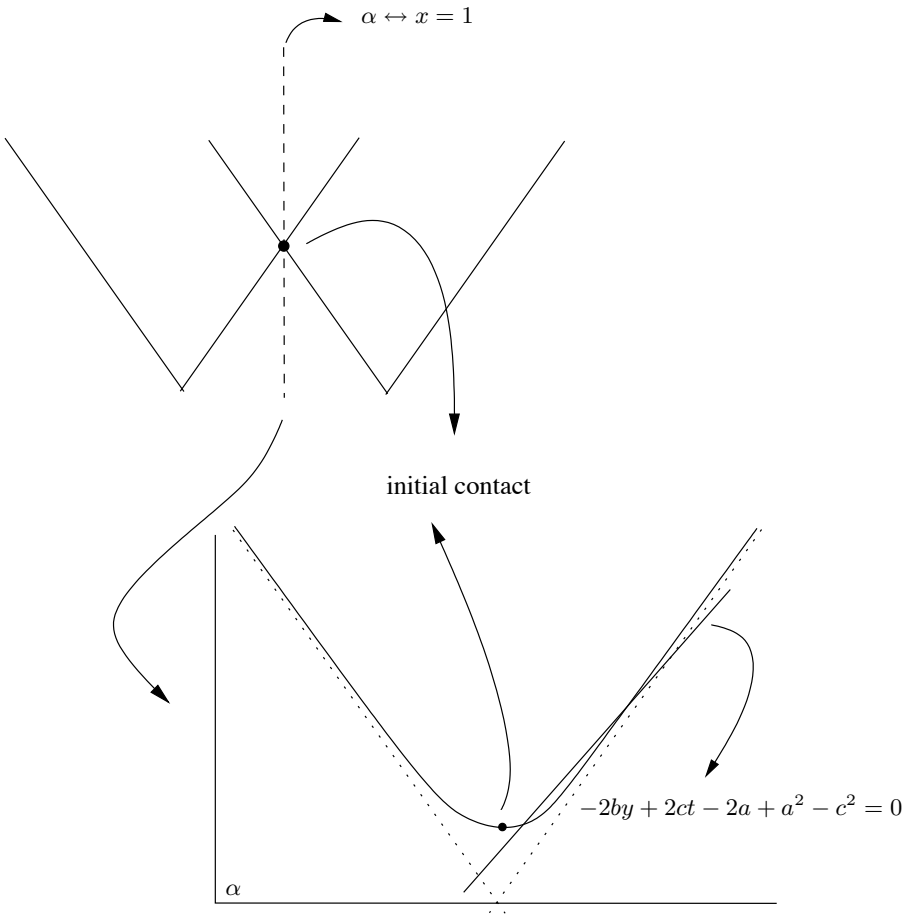


Figure 4.9: The initial contact cut out.

is satisfiable.

We first observe that an intersection between space-time prisms can be classified into three, mutually exclusive, cases. The three cases then are:

- **Case I:** an apex of one space-time prism is contained in the other, i.e.,

$$\tau\mathcal{P}_1 \cap \mathcal{P}_2 \neq \emptyset \text{ or } \mathcal{P}_1 \cap \tau\mathcal{P}_2 \neq \emptyset;$$

- **Case II:** not **Case I**, but the rim of one space-time prism intersects the mantel of the other, i.e.,

$$\rho\mathcal{P}_1 \cap \partial\mathcal{P}_2 \neq \emptyset \text{ or } \rho\mathcal{P}_2 \cap \partial\mathcal{P}_1 \neq \emptyset;$$

- **Case III:** not **Case I** and not **Case II** and the initial contact of the upper or lower cones is in the intersection of the space-time prisms, i.e.,

$$\mathsf{IC}(\mathsf{C}_1^-, \mathsf{C}_2^-) \subset \mathcal{P}_1 \cap \mathcal{P}_2 \text{ or } \mathsf{IC}(\mathsf{C}_1^+, \mathsf{C}_2^+) \subset \mathcal{P}_1 \cap \mathcal{P}_2.$$

If none of these three cases occur then the space-time prisms do not intersect, as we show in the correctness proof below. First, we give the following geometric lemma.

Lemma 4.1. *If* $\mathcal{P}_1 \cap \mathcal{P}_2 \neq \emptyset$, $\tau\mathcal{P}_1 \cap \mathcal{P}_2 = \emptyset$ *and* $\tau\mathcal{P}_2 \cap \mathcal{P}_1 = \emptyset$, *then* $\partial\mathcal{P}_1 \cap \partial\mathcal{P}_2 \neq \emptyset$.

Proof. From the assumptions, we know there is a point p_1 in $\mathcal{P}_2$, in particular e.g., an apex of $\mathcal{P}_2$, that is not in $\mathcal{P}_1$. Also, there is a point p_2 that is in $\mathcal{P}_2$ and in $\mathcal{P}_1$. The line segment bounded by p_1 and p_2 lies in $\mathcal{P}_2$, since $\mathcal{P}_2$ is convex. The line segment cuts the mantel of $\mathcal{P}_1$ since p_2 is inside $\mathcal{P}_1$ and p_1 is not. Let p be this point where the segment bounded by p_1 and p_2 intersects $\partial\mathcal{P}_1$. This point lies either on the upper-half space-time prism $\mathcal{P}_1^+$ or on the bottom-half space-time prism $\mathcal{P}_1^-$. Let r be the apex of this half space-time prism. Since p is inside $\mathcal{P}_2$ and r is not, the line segment bounded by p and r must cut $\partial\mathcal{P}_2$ in a point q. This point lies of course on $\partial\mathcal{P}_2$ and on $\partial\mathcal{P}_1$ since the line segment bounded by p and r is a part of $\partial\mathcal{P}_1$. Hence their mantels must have a non-empty intersection if the space-time prisms have a non-empty intersection and neither space-time prism contains the apexes of the other. □

Now, we show that if $\mathcal{P}_1$ and $\mathcal{P}_2$ intersect and neither Case I, nor Case II occurs, then Case III occurs.

Theorem 4.2. *If* $\mathcal{P}_1 \cap \mathcal{P}_2 \neq \emptyset$, $\tau\mathcal{P}_1 \cap \mathcal{P}_2 = \emptyset$, $\mathcal{P}_1 \cap \tau\mathcal{P}_2 = \emptyset$, $\rho\mathcal{P}_1 \cap \partial\mathcal{P}_2 = \emptyset$ *and* $\rho\mathcal{P}_2 \cap \partial\mathcal{P}_1 = \emptyset$, *then* $\mathsf{IC}(\mathsf{C}_1^-, \mathsf{C}_2^-) \subset \mathcal{P}_1 \cap \mathcal{P}_2$ *or* $\mathsf{IC}(\mathsf{C}_1^+, \mathsf{C}_2^+) \subset \mathcal{P}_1 \cap \mathcal{P}_2$.

Proof. Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be the space-time prisms $\mathcal{P}_1 = \mathcal{P}(t_1, x_1, y_1, t_2, x_2, y_2, v_1)$ and $\mathcal{P}_2 = \mathcal{P}(t_3, x_3, y_3, t_4, x_4, y_4, v_2)$.

Let us assume that the hypotheses of the statement of the theorem is true. It is sufficient to prove that either $\mathsf{C}_1^- \cap \mathsf{C}_2^- \subset \mathcal{P}_1^- \cap \mathcal{P}_2^-$ or $\mathsf{C}_1^+ \cap \mathsf{C}_2^+ \subset \mathcal{P}_1^+ \cap \mathcal{P}_2^+$. We will split the proof in two cases. From the fourth and fifth hypotheses it follows that either (1) $\rho\mathcal{P}_1 \subset \mathcal{P}_2$ or $\rho\mathcal{P}_2 \subset \mathcal{P}_1$; or (2) $\rho\mathcal{P}_1 \cap \mathcal{P}_2 = \emptyset$ and $\rho\mathcal{P}_2 \cap \mathcal{P}_1 = \emptyset$.

Case (1): We assume $\rho\mathcal{P}_2 \subset \mathcal{P}_1$ (the case $\rho\mathcal{P}_1 \subset \mathcal{P}_2$ is completely analogous). We prove $\mathsf{C}_1^- \cap \mathsf{C}_2^- \subset \mathcal{P}_1^- \cap \mathcal{P}_2^-$ (the case for upper cones is completely analogous). The following argument is illustrated in Figure 4.10.

Since $\rho\mathcal{P}_2 \subset \mathcal{P}_1$, we know that $\rho\mathcal{P}_2$ is inside C_1^-, and (t_3, x_3, y_3) is outside. We can show that $v_2 < v_1$. Consider the plane spanned by the two axis of

symmetry of both C_1^- and C_2^-. Both C_1^- and C_2^- intersect this plane in two half lines each. Moreover, we know that C_1^- intersects the axis of symmetry of C_2^-. Let t_0 be the moment at which this happens. Obviously $t_0 > t_1$, but we also know $t_0 > t_3$ since (t_3, x_3, y_3) is outside C_1^-. We have that $v_1(t_0 - t_1) = \sqrt{(x_1 - x_3)^2 + (y_1 - y_3)^2}$. Since $\rho\mathcal{P}_2$ is inside C_1^- and (t_3, x_3, y_3) is outside, this means both half lines from C_2^- intersect the half lines from C_1^-. Let t_0' and t_0'' be the moments in time at which this happens and let $t_0' > t_0''$. We have again that $t_0' > t_1$ and $t_0' > t_3$. Then $v_1(t_0' - t_1) = \sqrt{(x_1 - x_3)^2 + (y_1 - y_3)^2} + v_2(t_0' - t_3)$ if and only if $v_1(t_0' - t_0) = v_2(t_0' - t_3)$. Since $t_0 > t_3$, we get $v_2 < v_1$. This is depicted in Figure 4.10.

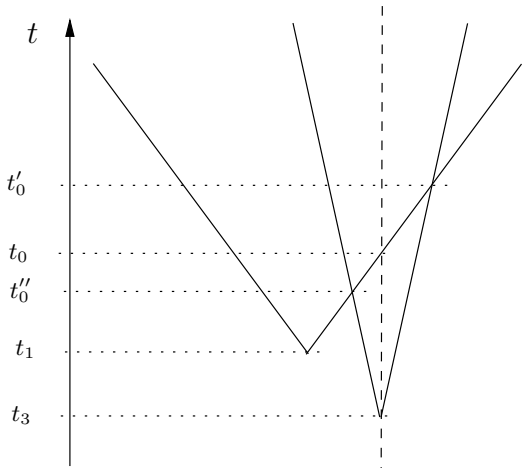


Figure 4.10: Illustration to the proof.

It follows that every straight half line starting in (t_3, x_3, y_3) on C_2^- intersects C_1^- between (t_3, x_3, y_3) and $\rho\mathcal{P}_2$, since $\rho\mathcal{P}_2$ is inside C_1^-, and (t_3, x_3, y_3) is outside. We also know that this line does not intersect C_1^- beyond $\rho\mathcal{P}_2$ since the cone C_2^- is entirely inside C_1^- beyond the rim $\rho\mathcal{P}_2$. Therefore, $\mathsf{C}_1^- \cap \mathsf{C}_2^- \subset \mathcal{P}_2^-$.

Clearly, $\mathcal{P}_2^-$ intersects $\mathcal{P}_1^-$ since it can not intersect $\mathcal{P}_1^+$. We know $\mathsf{C}_1^- \cap \partial\mathcal{P}_2^-$ is a closed continuous curve that lies entirely in C_1^-. This curve is also contained in $\mathcal{P}_1^-$. Indeed, if we assume this is not the case, then it intersects the plane in which $\rho\mathcal{P}_1$ lies, and hence it intersects $\rho\mathcal{P}_1$ itself, contradicting the assumption $\rho\mathcal{P}_1 \cap \partial\mathcal{P}_2 = \emptyset$.

Case (2): Now assume $\rho\mathcal{P}_1 \cap \mathcal{P}_2 = \emptyset$ and $\rho\mathcal{P}_2 \cap \mathcal{P}_1 = \emptyset$. Clearly, v_1 can not be equal to v_2, otherwise the depicted intersection can not occur. So suppose without loss of generality that $v_2 < v_1$. Now either $\mathcal{P}_2^-$ intersects both $\mathcal{P}_1^-$ and $\mathcal{P}_1^+$ or $\mathcal{P}_2^+$ intersects both $\mathcal{P}_1^-$ and $\mathcal{P}_1^+$. These cases are mutually exclusive because of the following. If $\mathcal{P}_2^+$ intersects $\mathcal{P}_1^+$, then $\rho\mathcal{P}_2$ is inside C_1^+, likewise if

$\mathcal{P}_2^-$ intersects $\mathcal{P}_1^-$, then $\rho\mathcal{P}_2$ is inside C_1^-. Hence $\rho\mathcal{P}_2 \subset \mathcal{P}_1$, which contradicts our hypothesis. If $\mathcal{P}_2^+$ intersects $\mathcal{P}_1^-$, then $\rho\mathcal{P}_2$ must be outside C_1^- and thus $\mathcal{P}_2^-$ must be as well, hence $\mathcal{P}_2^-$ intersects neither $\mathcal{P}_1^-$ nor $\mathcal{P}_1^+$. Likewise, if $\mathcal{P}_2^-$ intersects $\mathcal{P}_1^+$, then $\mathcal{P}_2^+$ can not intersect $\mathcal{P}_1$.

To prove that, "if $\mathcal{P}_2^-$ intersects $\mathcal{P}_1^-$, then it also intersects $\mathcal{P}_1^+$ and if $\mathcal{P}_2^-$ intersects $\mathcal{P}_1^+$, then it also intersects $\mathcal{P}_1^-$", we proceed as follows (the case for $\mathcal{P}_2^+$ is analogous). Suppose $\mathcal{P}_2^-$ intersects $\mathcal{P}_1^-$, then $\mathcal{P}_2^- \cap \mathcal{P}_1^- \subset \mathcal{P}_1$, but $\rho\mathcal{P}_2$ is outside $\mathcal{P}_1$, that means $\mathcal{P}_2^-$ must intersect $\mathcal{P}_1^+$ since it can not intersect $\mathcal{P}_1^-$ anymore. This is the "*what goes in must come out*"-principle. Likewise, suppose $\mathcal{P}_2^-$ intersects $\mathcal{P}_1^+$, then $\mathcal{P}_2^- \cap \mathcal{P}_1^+ \subset \mathcal{P}_1$, but (t_3, x_3, y_3) is outside $\mathcal{P}_1$, that means $\mathcal{P}_2^-$ must intersect $\mathcal{P}_1^-$ since it can not intersect $\mathcal{P}_1^-$ anymore.

So suppose now that $\mathcal{P}_2^-$ intersects both $\mathcal{P}_1^-$ and $\mathcal{P}_1^+$ (the case for $\mathcal{P}_2^+$ is completely analogous). If $\mathcal{P}_2^-$ intersects $\mathcal{P}_1^-$, that means $\rho\mathcal{P}_2$ is completely inside C_1^- and therefore that $\mathsf{C}_1^- \cap \mathsf{C}_2^- \subset \mathcal{P}_2^-$. We proceed like in the first case, we know that $\mathsf{C}_1^- \cap \mathcal{P}_2^-$ is a closed continuous curve. This curve lies entirely in C_1^-. If this curve is not entirely in $\mathcal{P}_1^-$ that means it intersects the plane in which $\rho\mathcal{P}_1$ lies, and hence intersects $\rho\mathcal{P}_1$ itself. But this is contradictory to the assumption that $\rho\mathcal{P}_1 \cap \partial\mathcal{P}_2 = \emptyset$. □

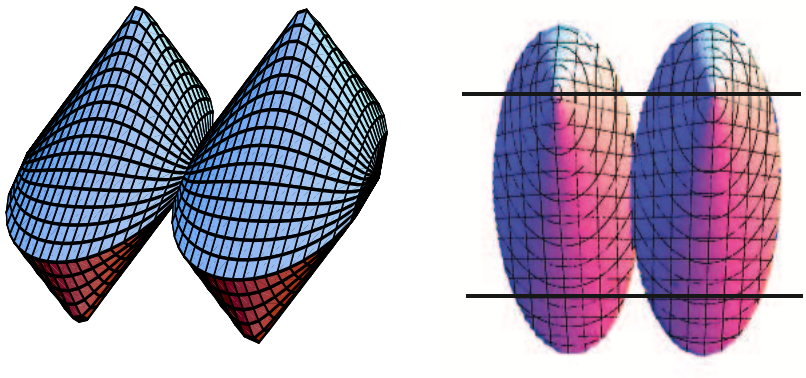


Figure 4.11: Case II is not redundant: $\mathcal{P}(0, 0, 0, 2, 0, 2, 1.9)$ and $\mathcal{P}(0, 3, 0, 2, 3, 2, 1.9)$ seen from the side and from the top.

In Theorem 4.2, we proved that if there is an intersection and neither rim cuts the other space-time prism's mantel and neither apex of a space-time prism is contained in the other then there must be an initial contact in the intersection. Visualizing how space-time prisms intersect might tempt one to think there is always an initial contact in the intersection. There exist counterexamples in which there is an intersection and no initial contact is in that intersection. That means case Case II is not redundant. This situation

is depicted in Figure 4.11. The space-time prisms are $\mathcal{P}(0, 0, 0, 2, 0, 2, 1.9)$ and $\mathcal{P}(0, 3, 0, 2, 3, 2, 1.9)$.

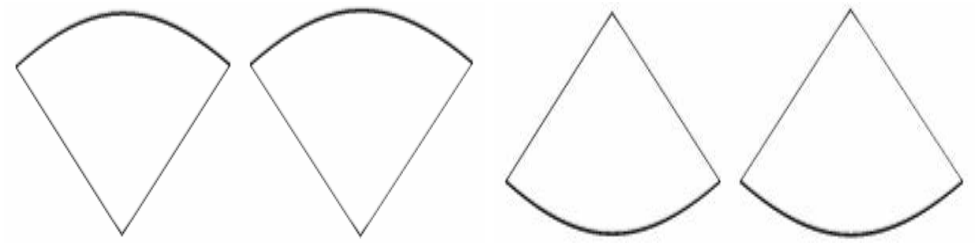


Figure 4.12: Intersection of Figure 4.11 with the plane $y = 0$ (left) and with the plane $y = 3$ (right).

It is clear that the initial contact of the bottom cones lies in the plane spanned by the axis of symmetry of those bottom cones, in this case this is the plane $y = 0$. The intersection of Figure 4.11 can be seen in Figure 4.12 on the left, where the two space-time prisms clearly have no intersection and thus no initial contact in the intersection.

In the case of the upper cones the initial contact must lie in the plane $y = 3$. The intersection of Figure 4.11 can be seen in Figure 4.12 on the right, where the two space-time prisms clearly have no intersection and there is again no initial contact in the intersection.

This concludes the outline.

Now we translate the three cases, Case I, Case II and Case III, into analytical formulas.

4.2.5.3 A formula for Case I

In Case **(I)**, we verify whether $\tau\mathcal{P}_1 \cap \mathcal{P}_2 \neq \emptyset$ or $\mathcal{P}_1 \cap \tau\mathcal{P}_2 \neq \emptyset$. To check if that is the case we merely need to verify if one of the apexes satisfies the set of equations of the other space-time prism. In this way we obtain

$$\begin{aligned}\Phi_{\mathbf{I}}\left(t_1, x_1, y_1, t_2, x_2, y_2, v_1, t_3, x_3, y_3, t_4, x_4, y_4, v_2\right) := \\ \psi_{\mathcal{P}}\left(t_3, x_3, y_3, t_1, x_1, y_1, t_2, x_2, y_2, v_1\right) \ \vee \\ \psi_{\mathcal{P}}\left(t_4, x_4, y_4, t_1, x_1, y_1, t_2, x_2, y_2, v_1\right) \ \vee \\ \psi_{\mathcal{P}}\left(t_1, x_1, y_1, t_3, x_3, y_3, t_4, x_4, y_4, v_2\right) \ \vee \\ \psi_{\mathcal{P}}\left(t_2, x_2, y_2, t_3, x_3, y_3, t_4, x_4, y_4, v_2\right).\end{aligned}$$

For the following sections, Cases II and III, we assume that the apex sets of the space-time prisms are not singletons, i.e., $t_1 < t_2$ and $t_3 < t_4$.

4.2.5.4 A formula for Case II

Now, let us assume that $\Phi_{\mathbf{I}}$ from the previous section failed. We note that we can always apply a speed-preserving transformation to $\mathbf{R} \times \mathbf{R}^2$ to obtain easier coordinates. We can always find a transformation such that $(t'_1, x'_1, y'_1) = (0, 0, 0)$ and that the line-segment connecting (t'_1, x'_1, y'_1) and (t'_2, x'_2, y'_2) is perpendicular to the y-axis, i.e., $y'_2 = 0$. This transformation is a composition of a translation of $\mathbf{R} \times \mathbf{R}^2$, a spatial rotation of $\mathbf{R}^2$ and a scaling of $\mathbf{R} \times \mathbf{R}^2$, see Chapter 3. Let the coordinates without a prime be the original set, and let coordinates with a prime be the image of the same coordinates without a prime under this transformation. We note that we do not need to transform back because the query is invariant under such transformations, as we proved in Chapter 3. The following formula returns the transformed coordinates (t', x', y') of (t, x, y) given the points (t_1, x_1, y_1) and (t_2, x_2, y_2):

$$
\begin{aligned}
&\varphi_A(t_1, x_1, y_1, t_2, x_2, y_2, t, x, y, t', x', y') := \\
&\qquad \big(y_2 = y_1 \ \wedge \ t' = (t - t_1) \ \wedge \ x' = (x - x_1) \ \wedge \ y' = (y - y_1)\big) \\
&\qquad\quad \vee \ \Big(y_2 \neq y_1 \ \wedge \ t' = (t - t_1)\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \\
&\qquad\qquad \wedge \ x' = (x - x_1)(x_2 - x_1) + (y - y_1)(y_2 - y_1) \\
&\qquad\qquad\qquad\qquad \wedge \ y' = (x - x_1)(y_1 - y_2) + (y - y_1)(x_2 - x_1)\Big) .
\end{aligned}
$$

The translation is over the vector $(-t_1, -x_1, -y_1)$, the rotation over minus the angle that $(t_2 - t_1, x_2 - x_1, y_2 - y_1)$ makes with the x-axis, and a scaling by a factor $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$. We note that the rotation and scaling only need to occur if y_2 is not already in place, i.e., if $y_2 \neq y_1$.

The formula $\chi(t'_1, x'_1, y'_1, t_1, x_1, y_1, t'_2, x'_2, y'_2, t_2, x_2, y_2, t'_3, x'_3, y'_3, t_3, x_3, y_3, t'_4, x'_4, y'_4, t_4, x_4, y_4)$ is an abbreviation for

$$
\begin{aligned}
&\varphi_A(t_1, x_1, y_1, t_2, x_2, y_2, t_1, x_1, y_1, t'_1, x'_1, y'_1) \ \wedge \\
&\qquad \varphi_A(t_1, x_1, y_1, t_2, x_2, y_2, t_2, x_2, y_2, t'_2, x'_2, y'_2) \ \wedge \\
&\qquad \varphi_A(t_1, x_1, y_1, t_2, x_2, y_2, t_3, x_3, y_3, t'_3, x'_3, y'_3) \ \wedge \\
&\qquad\qquad \varphi_A(t_1, x_1, y_1, t_2, x_2, y_2, t_4, x_4, y_4, t'_4, x'_4, y'_4).
\end{aligned}
$$

This transformation yields some simple equations for the rim $\rho\mathcal{P}_1$:

$$
\begin{cases}
x^2 + y^2 = t^2 v_1^2 \\
2x(-x'_2) + x'^2_2 = v_1^2(2t(-t'_2) + t'^2_2) \\
0 \leq t \leq t'_2 \ .
\end{cases}
$$

Not only that, but with these equations we can deduce a simple parametrization in the x-coordinate for the rim,

$$\begin{cases} t = \frac{2xx'_2 - x'^2_2 + v_1^2 t'^2_2}{2v_1^2 t'_2} \\ y = \pm\sqrt{v_1^2 \left(\frac{2xx'_2 - x'^2_2 + v_1^2 t'^2_2}{2v_1^2 t'_2}\right)^2 - x^2} \\ 0 \le t \le t'_2 \ . \end{cases}$$

We remark that this implies $t'_2 \neq 0$ and $v_1 \neq 0$. If $t'_2 = 0$, then $\mathcal{P}_1$ is a point, hence degenerate. If $v_1 = 0$, then $\mathcal{P}_1$ is a line segment, and again degenerate. Next we will inject these parameterizations in the constraints for $\partial\mathcal{P}_2^+$ and $\partial\mathcal{P}_2^-$ separately. The constraints for $\partial\mathcal{P}_2^-$ are

$$\begin{cases} (x - x'_3)^2 + (y - y'_3)^2 = (t - t'_3)^2 v_2^2 \\ 2x(x'_3 - x'_4) + x'^2_4 - x'^2_3 + 2y(y'_3 - y'_4) + y'^2_4 - y'^2_3 \\ \qquad\qquad \le v_2^2 \left(2t(t'_3 - t'_4) + t'^2_4 - t'^2_3\right) \\ t'_3 \le t \le t'_4 \ . \end{cases}$$

We will explain how to proceed to compute the intersection with $\partial\mathcal{P}_2^-$ and simply reuse formulas for intersection with $\partial\mathcal{P}_2^+$. We insert our expressions for x and y in the first equation. This is equivalent to computing intersections of $\rho\mathcal{P}_1$ with C_2^- and gives

$$\left(x - x'_3\right)^2 + \left(\pm\sqrt{v_1^2 \left(\frac{2xx'_2 - x'^2_2 + v_1^2 t'^2_2}{2v_1^2 t'_2}\right)^2 - x^2} - y'_3\right)^2 = \left(\frac{2xx'_2 - x'^2_2 + v_1^2 t'^2_2}{2v_1^2 t'_2} - t'_3\right)^2 v_2^2,$$

or equivalently

$$\pm 2y'_3\sqrt{v_1^2\left(2xx'_2 - x'^2_2 + v_1^2 t'^2_2\right)^2 - \left(2v_1^2 t'_2\right)^2 x^2} = \left(2xx'_2 - x'^2_2 + v_1^2 t'^2_2 - \left(2v_1^2 t'_2\right) t'_3\right)^2 v_2^2 - \left(2v_1^2 t'_2\right)^2 (x - x'_3)^2 - \left(2v_1^2 t'_2\right)^2 y'^2_3 - \left(v_1^2\left(2xx'_2 - x'^2_2 + v_1^2 t'^2_2\right)^2 - \left(2v_1^2 t'_2\right)^2 x^2\right)$$

or equivalently

$$\pm v_1 2y'_3\sqrt{x^2 4\left(x'^2_2 - v_1^2 t'^2_2\right) + x4x'^2_2\left(v_1^2 t'^2_2 - x'^2_2\right) + \left(v_1^2 t'^2_2 - x'^2_2\right)^2} = x^2 4x'^2_2 v_2^2 - x^2 4x'^2_2 v_1^2 + x4\big(- x'^2_2 v_2^2\left(-x'^2_2 + v_1^2 t'^2_2 - 4v_1^4 t'^2_2 t'_3\right)$$

$$+ 2v_1^4 t_2'^2 x_3' + v_1^2 x_2' \left(v_1^2 t_2'^2 - x_2'^2\right)) + \big(v_2^2 \left(-x_2'^2 + v_1^2 t_2'^2 - 4v_1^4 t_2'^2 t_3'\right)^2 - 4v_1^4 t_2'^2 \left(x_3'^2 + y_3'^2\right) - v_1^4 \left(-x_2'^2 + v_1^2 t_2'^2\right)\big).$$

By squaring left and right hand in this last expression, we rid ourselves of the square root and obtain the following polynomial equation of degree four. Squaring may create new solutions, so to ensure we only get useful solutions, we have to add the condition that the square root exists. This is the case if and only if

$$\phi_{\surd}(x, t_2', x_2', v_1) := x^2 4 \left(x_2'^2 - v_1^2 t_2'^2\right) + x 4 x_2'^2 \left(v_1^2 t_2'^2 - x_2'^2\right) + \left(v_1^2 t_2'^2 - x_2'^2\right)^2 \geq 0$$

is satisfied.

We notice that if $\mathcal{P}_1$ is degenerate, i.e., $x_2'^2 = v_1^2 t_2'^2$, then the square root vanishes and the polynomial in ϕ_4 is the square of a polynomial of degree two, yielding to at most two roots and intersection points as we expect. The case were $v_1 = 0$ is captured by the formula in the next section, that is why we leave that case out here and demand that $v_1 \neq 0$. So the following still works if one or both space-time prisms is degenerate:

$$\begin{aligned}
&\phi_4(x, t_2', x_2', v_1, t_3', x_3', y_3', v_2) := \\
&\exists a \exists b \exists c \exists d \exists e \Big(ax^4 + bx^3 + cx^2 + dx + e = 0 \wedge\ a = \left(4x_2'^2 \left(v_2^2 - v_1^2\right)\right)^2 \\
&\wedge\ b = -32 x_2'^4 v_2^2 \left(v_2^2 - v_1^2\right) \left(-x_2'^2 + v_1^2 t_2'^2 - 4v_1^4 t_2'^2 t_3' + 2v_1^4 t_2'^2 x_3'\right. \\
&\left. + v_1^2 x_2' \left(v_1^2 t_2'^2 - x_2'^2\right)\right) \wedge\ c = 8 \left(x_2'^2 - v_1^2 t_2'^2\right) \left(-4v_1^4 t_2'^2 \left(x_3'^2 + y_3'^2\right)\right. \\
&\left. + v_2^2 \left(-x_2'^2 + v_1^2 t_2'^2 - 4v_1^4 t_2'^2 t_3'\right)^2 - v_1^4 \left(-x_2'^2 + v_1^2 t_2'^2\right)\right) + \\
&\left(2v_1 y_3'\right)^2 \left(x_2'^2 - v_1^2 t_2'^2\right) + \big(4\big(- x_2'^2 v_2^2 \left(-x_2'^2 + v_1^2 t_2'^2 - 4v_1^4 t_2'^2 t_3'\right) \\
&+ 2v_1^4 t_2'^2 x_3' + v_1^2 x_2' \left(v_1^2 t_2'^2 - x_2'^2\right)\big)\big)^2 \wedge\ d = 8\big(2v_1^4 t_2'^2 x_3' - \\
&x_2'^2 v_2^2 \left(-x_2'^2 + v_1^2 t_2'^2 - 4v_1^4 t_2'^2 t_3'\right) + v_1^2 x_2' \left(v_1^2 t_2'^2 - x_2'^2\right)\big) \\
&\left(v_2^2 \left(-x_2'^2 + v_1^2 t_2'^2 - 4v_1^4 t_2'^2 t_3'\right)^2 - 4v_1^4 t_2'^2 \left(x_3'^2 + y_3'^2\right)\right. \\
&\left. - v_1^4 \left(-x_2'^2 + v_1^2 t_2'^2\right)\right) + \left(2v_1 y_3'\right)^2 \left(4x_2'^2 \left(v_1^2 t_2'^2 - x_2'^2\right)\right) \\
&\wedge\ e = \left(2v_1 y_3'\right)^2 \left(v_1^2 t_2'^2 - x_2'^2\right)^2 + \big(- 4v_1^4 t_2'^2 \left(x_3'^2 + y_3'^2\right) \\
&+ v_2^2 \left(-x_2'^2 + v_1^2 t_2'^2 - 4v_1^4 t_2'^2 t_3'\right)^2 - v_1^4 \left(-x_2'^2 + v_1^2 t_2'^2\right)\big)^2 \Big).
\end{aligned}$$

The quantifiers, we introduced here, are only in place for esthetical considerations and can be eliminated by direct substitution.

We note that if $v_1 = v_2$, we get polynomials of degree merely two. This can be solved in an exact manner using nested square roots (or Maple if you want). This gives us at most four values for x. Let

$$\phi_{roots}(x_a, x_b, x_c, x_d, t'_2, x'_2, v_1, t'_3, x'_3, y'_3, v_2)$$

be a formula that returns all four real roots, if they exist, that satisfy both $\phi_4(x, t'_2, x'_2, v_1, t'_3, x'_3, y'_3, v_2)$ and $\phi_{\sqrt{\ }}(x, t'_2, x'_2, v_1)$. We substitute these values in the parameter equations of $\rho\mathcal{P}_1$. By substituting these in the last equation above, we can determine the sign of the square root we need to take for y. A point (t, x, y) satisfies the following formula is a point on $\rho\mathcal{P}_1$, but instead of using the square root for y, we use an expression from above to get the correct sign for the square root if $y'_3 \neq 0$. If $y'_3 = 0$ we have to use the square root expression and then it does not matter which sign the square root has; we need both:

$$\begin{aligned}
&\psi_\rho(t, x, y, t'_2, x'_2, v_1, t'_3, x'_3, y'_3, v_2) := \\
&\quad \left(y'_3 \neq 0 \ \wedge \ t\left(2v_1^2 t'_2\right) = 2xx'_2 - x_2'^2 + v_1^2 t_2'^2 \wedge \ 2y'_3\left(2v_1^2 t'_2\right) y = \right. \\
&\qquad \left(2xx'_2 - x_2'^2 + v_1^2 t_2'^2 - \left(2v_1^2 t'_2\right) t'_3\right)^2 v_2^2 - \left(2v_1^2 t'_2\right)^2 (x - x'_3)^2 \\
&\qquad - \left(2v_1^2 t'_2\right)^2 y_3'^2 - \left(v_1^2\left(2xx'_2 - x_2'^2 + v_1^2 t_2'^2\right)^2 - \left(2v_1^2 t'_2\right)^2 x^2\right) \\
&\quad \wedge \ \ 0 \leq t \leq t'_2 \ ) \vee \left(y'_3 = 0 \ \wedge \ t\left(2v_1^2 t'_2\right) = 2xx'_2 - x_2'^2 + v_1^2 t_2'^2\right. \\
&\quad \wedge \ 0 \leq t \leq t'_2 \ \wedge \ \left(2v_1^2 t'_2\right)^2 y^2 = \left(2xx'_2 - x_2'^2 + v_1^2 t_2'^2\right)^2 - \left(2v_1^2 t'_2\right)^2 x^2\Big).
\end{aligned}$$

The four roots give us four time-space points on $\rho\mathcal{P}_1 \cap \mathsf{C}_2^-$. In order for these points (t, x, y) to be in $\rho\mathcal{P}_1 \cap \partial\mathcal{P}_2^-$, they need to satisfy

$$\begin{aligned}
&\psi_-(t, x, y, t'_3, x'_3, y'_3, t'_4, x'_4, y'_4, v_2) := \\
&\qquad 2x(x'_3 - x'_4) + x_4'^2 - x_3'^2 + 2y(y'_3 - y'_4) + y_4'^2 - y_3'^2 \\
&\qquad\qquad \leq v_2^2\left(2t(t'_3 - t'_4) + t_4'^2 - t_3'^2\right).
\end{aligned}$$

This formula returns *true* if (t, x, y) lies in the same half space as the bottom-half space-time prism.

The formula ψ_+ returns *true* if (t, x, y) lies in the same half space as the upper-half space-time prism, i.e.,

$$\psi_+(t, x, y, t'_3, x'_3, y'_3, t'_4, x'_4, y'_4, v_2) := \psi_-(t, x, y, t'_4, x'_4, y'_4, t'_3, x'_3, y'_3, v_2).$$

By combining $\psi_\rho(t, x, y, t'_2, x'_2, v_1, t'_3, x'_3, y'_3, v_2)$ and $\psi_-(t, x, y, \hat{t}, \hat{x}, \hat{y}, \tilde{t}, \tilde{x}, \tilde{y}, v)$, we get a formula that decides the emptiness of the intersection $\rho\mathcal{P}_1 \cap \partial\mathcal{P}_2^-$ in terms of a parameter x:

$$\psi_{\rho\cap\partial^\pm}(x, t'_2, x'_2, v_1, t'_3, x'_3, y'_3, v_2, \hat{t}, \hat{x}, \hat{y}, \tilde{t}, \tilde{x}, \tilde{y}, v) :=$$

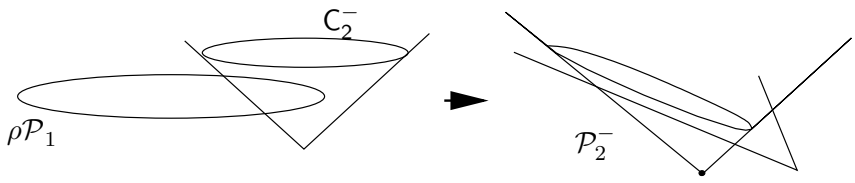


Figure 4.13: The rim intersects the cone and solutions are verified in a half-space.

$$\begin{aligned}
\exists y\ \Big(y_3' = 0\ \wedge\ y^2\left(2v_1^2t_2'\right)^2 = \left(2xx_2' - x_2'^2 + v_1^2t_2'^2\right)^2 v_1^2 - \left(2v_1^2t_2'\right)^2 x^2\Big)\\
\vee\ \Big(y_3' \neq 0 \wedge\ 2y_3'\left(2v_1^2t_2'\right) y = \left(2xx_2' - x_2'^2 + v_1^2t_2'^2 - \left(2v_1^2t_2'\right) t_3'\right)^2 v_2^2 -\\
\left(2v_1^2t_2'\right)^2 (x - x_3')^2 - \Big(v_1^2\left(2xx_2' - x_2'^2 + v_1^2t_2'^2\right)^2 - \left(2v_1^2t_2'\right)^2 x^2\Big)\\
- \left(2v_1^2t_2'\right)^2 y_3'^2\Big) \wedge \big(2x(\hat{x} - \tilde{x}) + \tilde{x}^2 - \hat{x}^2 + 2y(\hat{y} - \tilde{y}) + \tilde{y}^2\\
- \hat{y}^2\big)\left(2v_1^2t_2'\right) \leq v^2\big(2\left(2v_1^2t_2'\right)\left(2xx_2' - x_2'^2 + v_1^2t_2'^2\right)(\hat{t} - \tilde{t})\\
+ \left(2v_1^2t_2'\right)\left(\tilde{t}^2 - \hat{t}^2\right)\big)\ \wedge\ 0 \leq t_2'\left(2xx_2' - x_2'^2 + v_1^2t_2'^2\right) \leq 2v_1^2t_2'^3\\
\wedge\ \big(\hat{t}\left(2v_1^2t_2'^2\right) \leq t_2'\left(2xx_2' - x_2'^2 + v_1^2t_2'^2\right) \leq \tilde{t}\left(2v_1^2t_2'^2\right)\\
\vee\ \tilde{t}\left(2v_1^2t_2'^2\right) \leq t_2'\left(2xx_2' - x_2'^2 + v_1^2t_2'^2\right) \leq \hat{t}\left(2v_1^2t_2'^2\right)\big)\Big).
\end{aligned}$$

We are ready now to construct the formula that decides if $\rho\mathcal{P}_1$ and $\mathcal{P}_2^-$ have a non-empty intersection:

$$\begin{aligned}
&\varphi_{\rho_1\cap\partial_2^-}(t_2', x_2', v_1, t_3', x_3', y_3', t_4', x_4', y_4', v_2) :=\\
&\quad \exists x\exists x_a\exists x_b\exists x_c\exists x_d\ \big(\ \phi_{roots}(x_a, x_b, x_c, x_d, t_2', x_2', v_1, t_3', x_3', y_3', v_2)\\
&\qquad \wedge\ (x = x_a\ \vee\ x = x_b\ \vee\ x = x_c\ \vee\ x = x_d)\\
&\qquad \wedge\ \psi_{\rho\cap\partial^\pm}(x, t_2', x_2', v_1, t_3', x_3', y_3', v_2, t_3', x_3', y_3', t_4', x_4', y_4', v_2')\big).
\end{aligned}$$

The formula that decides if $\rho\mathcal{P}_1$ intersects $\partial\mathcal{P}_2^+$ looks strikingly similar:

$$\begin{aligned}
&\varphi_{\rho_1\cap\partial_2^+}(t_2', x_2', v_1, t_3', x_3', y_3', t_4', x_4', y_4', v_2) :=\\
&\quad \exists x\exists x_a\exists x_b\exists x_c\exists x_d\ \big(\phi_{roots}(x_a, x_b, x_c, x_d, t_2', x_2', v_1, t_4', x_4', y_4', v_2)\\
&\qquad \wedge\ (x = x_a\ \vee\ x = x_b\ \vee\ x = x_c\ \vee\ x = x_d)\\
&\qquad \wedge\ \psi_{\rho\cap\partial^\pm}(x, t_2', x_2', v_1, t_4', x_4', y_4', v_2, t_4', x_4', y_4', t_3', x_3', y_3', v_2')\big).
\end{aligned}$$

The quantifiers introduced here can also be eliminated in a straightforward manner. Notice that ϕ_{roots} acts as a function rather than a formula that inputs $(t_2', x_2', v_1, t_4', x_4', y_4', v_2)$ to construct a polynomial of degree four and returns the four roots (x_a, x_b, x_c, x_d), if they exist, of that polynomial. The existential

quantifier for the variable x is used to cycle through those roots to see if any of them identifies a witness to the alibi query. Finally, we are ready to present the formula for Case II:

$$\begin{aligned}
&\Phi_{\mathbf{II}}\,(t_1,x_1,y_1,t_2,x_2,y_2,v_1,t_3,x_3,y_3,t_4,x_4,y_4,v_2) :=\\
&\qquad \neg\ \Phi_{\mathbf{I}}\,(t_1,x_1,y_1,t_2,x_2,y_2,v_1,t_3,x_3,y_3,t_4,x_4,y_4,v_2)\ \wedge\\
&\quad (v_1>0\ \wedge\ v_2>0)\ \wedge\ \exists t_1'\exists x_1'\exists y_1'\exists t_2'\exists x_2'\exists y_2'\exists t_3'\exists x_3'\exists y_3'\exists t_4'\exists x_4'\exists y_4'\\
&\qquad \big(\chi(t_1',x_1',y_1',t_1,x_1,y_1,t_2',x_2',y_2',t_2,x_2,y_2,t_3',x_3',y_3',t_3,x_3,y_3,\\
&\quad t_4',x_4',y_4',t_4,x_4,y_4)\wedge\big(\varphi_{\rho_1\cap\partial_2^-}(t_2',x_2',v_1,t_3',x_3',y_3',t_4',x_4',y_4',v_2')\ \vee\\
&\quad \varphi_{\rho_1\cap\partial_2^+}(t_2',x_2',v_1,t_3',x_3',y_3',t_4',x_4',y_4',v_2)\big)\ \vee\ \chi(t_3',x_3',y_3',t_3,x_3,y_3,\\
&\qquad t_4',x_4',y_4',t_4,x_4,y_4,t_1',x_1',y_1',t_1,x_1,y_1,t_2',x_2',y_2',t_2,x_2,y_2)\\
&\qquad\qquad \wedge\ \big(\varphi_{\rho_1\cap\partial_2^-}(t_3',x_3',v_2,t_1',x_1',y_1',t_2',x_2',y_2',v_1')\\
&\qquad\qquad\qquad \vee\ \varphi_{\rho_1\cap\partial_2^+}(t_3',x_3',v_2,t_1',x_1',y_1',t_2',x_2',y_2',v_1')\big)\big).
\end{aligned}$$

The reader may notice that a lot of quantifiers have been introduced in the formula above. These quantifiers are merely there to introduce easier coordinates and can be straightforwardly computed (and eliminated) by the formula χ and hence the formula $\varphi_A(t_1,x_1,y_1,t_2,x_2,y_2,t,x,y,t',x',y')$. The latter actually acts like a function, parameterized by $(t_1,x_1,y_1,t_2,x_2,y_2)$, that has input (t,x,y) and returns (t',x',y').

4.2.5.5 A formula for Case III

Here, we assume that both $\varphi_{\mathbf{I}}$ and $\varphi_{\mathbf{II}}$ fail. So, there is no apex contained in the other space-time prism and neither rim cuts the mantel of the other space-time prism.

As we proved in Theorem 4.2, the intersection of two half space-time prisms will reduce to the intersection of two cones and that means there is an initial contact that is part of the intersection. To verify if this is the case, we compute the two initial contacts and verify if they are effectively part of the intersection.

Using the expression for the initial contact $\mathsf{IC}(\mathsf{C}_1^-,\mathsf{C}_2^-)$, we computed in Section 4.2.4.2, we can construct a formula that decides if it is part of $\mathcal{P}_1^-\cap\mathcal{P}_2^-$. We will recycle the formulas ψ_- from the previous section to construct an expression without the need for extra variables. The following formula that returns *true* if $\psi_-(t_0,x_0,y_0,t',x',y',\hat{t},\hat{x},\hat{y},v)$ is satisfied where $\mathsf{IC}(\mathsf{C}_1^-,\mathsf{C}_2^-)=(t_0,x_0,y_0)$:

$$\begin{aligned}
&\phi_-(t_1,x_1,y_1,v_1,t_3,x_3,y_3,v_2,t',x',y',\hat{t},\hat{x},\hat{y},v) :=\\
&\quad\Big((x_1v_2+x_3v_1)\sqrt{(x_1-x_3)^2+(y_1-y_3)^2}+v_1\left((t_3-t_1)v_2\right)(x_3-x_1)\Big)
\end{aligned}$$

$$2(x' - \hat{x}) + 2(y' - \hat{y})\big((y_1 v_2 + y_3 v_1)\sqrt{(x_1 - x_3)^2 + (y_1 - y_3)^2} + v_1\left((t_3 - t_1)v_2\right)(y_3 - y_1)\big) + \sqrt{(x_1 - x_3)^2 + (y_1 - y_3)^2}\,(v_1 + v_2)\left(\hat{x}^2 - x'^2 + \hat{y}^2 - y'^2\right) \leq v^2\big(\left(\check{t}^2 - t'^2\right)(v_1 + v_2) + 2\left(\sqrt{(x_1 - x_3)^2 + (y_1 - y_3)^2} + t_1 v_1 + t_3 v_2\right)\left(t' - \hat{t}\right)\big)\sqrt{(x_1 - x_3)^2 + (y_1 - y_3)^2}.$$

The following formula expresses that the time coordinate t_0 of $\mathsf{IC}(\mathsf{C}_1^-, \mathsf{C}_2^-)$ satisfies the constraints $t' \leq t \leq t''$ and $\hat{t} \leq t \leq \check{t}$:

$$\psi_t\left(t_1, x_1, y_1, v_1, t_3, x_3, y_3, v_2, t', t'', \hat{t}, \check{t}\right) := t'(v_1 + v_2) \leq \sqrt{(x_1 - x_3)^2 + (y_1 - y_3)^2} + t_1 v_1 + t_3 v_2 \leq t''(v_1 + v_2) \wedge \hat{t}(v_1 + v_2) \leq \sqrt{(x_1 - x_3)^2 + (y_1 - y_3)^2} + t_1 v_1 + t_3 v_2 \leq \check{t}(v_1 + v_2).$$

Now, $\mathsf{IC}(\mathsf{C}_1^-, \mathsf{C}_2^-) \subset \mathcal{P}_1^- \cap \mathcal{P}_2^-$ if and only if $\psi_{\mathsf{IC}^-}(t_1, x_1, y_1, t_2, x_2, y_2, v_1, t_3, x_3, y_3, t_4, x_4, y_4, v_2)$ is satisfied where

$$\psi_{\mathsf{IC}^-}(t_1, x_1, y_1, t_2, x_2, y_2, v_1, t_3, x_3, y_3, t_4, x_4, y_4, v_2) := \psi_t(t_1, x_1, y_1, v_1, t_3, x_3, y_3, v_2, t_1, t_2, t_3, t_4) \wedge \phi_-(t_1, x_1, y_1, v_1, t_3, x_3, y_3, v_2, t_1, x_1, y_1, t_2, x_2, y_2, v_1) \wedge \phi_-(t_1, x_1, y_1, v_1, t_3, x_3, y_3, v_2, t_3, x_3, y_3, t_4, x_4, y_4, v_2)$$

and $\mathsf{IC}(\mathsf{C}_1^+, \mathsf{C}_2^+) \subset \mathcal{P}_1^+ \cap \mathcal{P}_2^+$ if and only if $\psi_{\mathsf{IC}^+}(t_1, x_1, y_1, t_2, x_2, y_2, v_1, t_3, x_3, y_3, t_4, x_4, y_4, v_2)$ is satisfied where

$$\psi_{\mathsf{IC}^+}(t_1, x_1, y_1, t_2, x_2, y_2, v_1, t_3, x_3, y_3, t_4, x_4, y_4, v_2) := \psi_t(t_2, x_2, y_2, v_1, t_4, x_4, y_4, v_2, t_1, t_2, t_3, t_4) \wedge \phi_-(t_2, x_2, y_2, v_1, t_4, x_4, y_4, v_2, t_2, x_2, y_2, t_1, x_1, y_1, v_1) \wedge \phi_-(t_2, x_2, y_2, v_1, t_4, x_4, y_4, v_2, t_4, x_4, y_4, t_3, x_3, y_3, v_2).$$

The formula that expresses the criterium for Case III then looks as follows:

$$\Phi_{\mathbf{III}}(t_1, x_1, y_1, t_2, x_2, y_2, v_1, t_3, x_3, y_3, t_4, x_4, y_4, v_2) := (v_1 + v_2 \neq 0) \wedge \neg\, \Phi_{\mathbf{I}}(t_1, x_1, y_1, t_2, x_2, y_2, v_1, t_3, x_3, y_3, t_4, x_4, y_4, v_2) \wedge (\psi_{\mathsf{IC}^-}(t_1, x_1, y_1, t_2, x_2, y_2, v_1, t_3, x_3, y_3, t_4, x_4, y_4, v_2) \vee \psi_{\mathsf{IC}^+}(t_1, x_1, y_1, t_2, x_2, y_2, v_1, t_3, x_3, y_3, t_4, x_4, y_4, v_2)).$$

4.2.5.6 The formula for the parametric alibi query

The final formula that decides if two space-time prisms, $\mathcal{P}_1 = \mathcal{P}(t_1, x_1, y_1, t_2, x_2, y_2, v_1)$ and $\mathcal{P}_2 = \mathcal{P}(t_3, x_3, y_3, t_4, x_4, y_4, v_2)$, do not intersect looks as follows

$$\psi_{alibi}(t_1, x_1, y_1, t_2, x_2, y_2, v_1, t_3, x_3, y_3, t_4, x_4, y_4, v_2) := \neg\left((t_1 < t_2 \wedge t_3 < t_4)\right.$$

$$\begin{aligned}\wedge\ & (\Phi_{\mathbf{III}}(t_1, x_1, y_1, t_2, x_2, y_2, v_1, t_3, x_3, y_3, t_4, x_4, y_4, v_2) \\ \vee\ & \Phi_{\mathbf{II}}(t_1, x_1, y_1, t_2, x_2, y_2, v_1, t_3, x_3, y_3, t_4, x_4, y_4, v_2)) \\ \vee\ & \Phi_{\mathbf{I}}(t_1, x_1, y_1, t_2, x_2, y_2, v_1, t_3, x_3, y_3, t_4, x_4, y_4, v_2))\,.\end{aligned}$$

4.2.6 Experiments

In this section, we compare our solution to the alibi query (using the formula given in Section 4.2.5.6) with the method of eliminating quantifiers of MATHEMATICA.

In the following table, it is clear that traditional quantifier elimination performs badly on the example space-time prisms. Its running times highly deviates from their average and range in the minutes. Whereas the method described in this chapter performs in running times that consistently only needs milliseconds or less. This shows our method is efficient and our claim, that it runs in milliseconds or less, holds.

For this first test of space-time prisms we chose to verify intersection of two oblique space-time prisms and the intersection of one oblique and one straight space-time prism. The space-time prisms that actually intersected had a remarkable low running time with MATHEMATICA.

The space-time prisms		The running times	
$\mathcal{P}_1$	$\mathcal{P}_2$	MATHEMATICA	ψ_{alibi}
$(0,0,0,2,0,2,1.9)$	$(0,3,0,2,3,2,2)$	0.656”	0.016”
$(0,0,0,2,0,2,1.9)$	$(0,4,0,2,4,2,2)$	324.453”	0.063”
$(0,0,0,2,0,2,1.9)$	$(0,3,0,2,3,0,2)$	0.438”	0.015”
$(0,0,0,2,0,2,1.9)$	$(0,4,0,2,4,0,2)$	475.719”	0.031”

The type of space-time prisms in this second test are as in the first. However, these space-time prisms all have overlapping time intervals unlike the first test, where the time intervals coincided.

The space-time prisms		The running times	
$\mathcal{P}_1$	$\mathcal{P}_2$	MATHEMATICA	ψ_{alibi}
$(0,0,0,2,0,2,1.9)$	$(1,3,0,3,3,2,2)$	63.375”	0.078”
$(0,0,0,2,0,2,1.9)$	$(1,4,0,3,4,2,2)$	59.485”	0.078”
$(0,0,0,2,0,2,1.9)$	$(1,3,0,3,3,0,2)$	29.734”	0.031”
$(0,0,0,2,0,2,1.9)$	$(1,4,0,3,4,0,2)$	27.281”	0.032”

The type of space-time prisms in this third test are as in the first. But this time the time intervals are completely disjoint. Note that the running times for MATHEMATICA are more consistent in this test and the previous one.

The space-time prisms		The running times	
$\mathcal{P}_1$	$\mathcal{P}_2$	MATHEMATICA	ψ_{alibi}
$(0,0,0,2,0,2,1.9)$	$(3,3,0,4,3,2,2)$	63.641"	0.046"
$(0,0,0,2,0,2,1.9)$	$(3,4,0,4,4,2,2)$	61.781"	0.016"
$(0,0,0,2,0,2,1.9)$	$(3,3,0,4,3,0,2)$	52.735"	0.031"
$(0,0,0,2,0,2,1.9)$	$(3,4,0,4,4,0,2)$	56.875"	0.046"

After conducting numerous experiments, we have to conclude that there is no consistent trend in the running times of the traditional method. The only trend that showed is that the time the traditional quantifier-elimination method takes is several orders of magnitude higher than our own method and that our method runs consistently under a tenth of a second.

4.2.7 The alibi query at a fixed moment in time

4.2.7.1 Introduction

In this section, we present another example where an ad-hoc solution prevails over the general quantifier-elimination methods. The problem is the following. As in the previous setting, we have lists of time stamped-locations of two moving objects and upper bounds on the object's speed between time stamps. We want to know if two objects may have met at a given moment in time.

For the remainder of this section, we reuse the assumptions from the previous section. We want to verify if the space-time prisms $\mathcal{P}_1 = \mathcal{P}(t_1, x_1, y_1, t_2, x_2, y_2, v_1)$ and $\mathcal{P}_2 = \mathcal{P}(t_3, x_3, y_3, t_4, x_4, y_4, v_2)$ intersect at a moment in time t_0. Moreover, we assume the space-time prisms are non-empty, i.e., $(x_2-x_1)^2+(y_2-y_1)^2 \leq (t_2-t_1)^2 v_1^2$ and $(x_4-x_3)^2+(y_4-y_3)^2 \leq (t_4-t_3)^2 v_2^2$ and that $t_1 \leq t_0 \leq t_2$ and $t_3 \leq t_0 \leq t_4$ are satisfied. This means we need to eliminate the quantifiers in

$$\begin{aligned}\exists x \exists y \big(& (x-x_1)^2+(y-y_1)^2 \leq v_1^2(t_0-t_1)^2 \\ & \wedge\ (x-x_2)^2+(y-y_2)^2 \leq v_1^2(t_0-t_2)^2 \\ & \wedge\ (x-x_3)^2+(y-y_3)^2 \leq v_2^2(t_0-t_3)^2 \\ & \wedge\ (x-x_4)^2+(y-y_4)^2 \leq v_2^2(t_0-t_4)^2 \big).\end{aligned}$$

Eliminating quantifiers gives us a formula that decide whether or not four discs have a non-empty intersection. For ease of notation we will use the following abbreviations: $(x,y) \in D_i$ if and only if $(x-x_i)^2+(y-y_i) \leq r_i^2$ and $(x,y) \in C_i$ if and only if $(x-x_i)^2+(y-y_i) = r_i^2$.

4.2.7.2 Main theorem

Using Helly's theorem [14] we can simplify the problem even more.

Theorem 4.3 (Helly)**.** *Let $X_1, \ldots, X_m$ be convex subsets of $\mathbf{R}^n$ where $m > n$. If any $n+1$ of these subsets have a non-empty intersection, then*

$$\bigcap_{i=1}^{m} X_i \neq \emptyset.$$

□

For the plane, this means we only need to find a quantifier free-formula that decides if three discs have a non-empty intersection. For the remainder of this section assume that we want to verify whether $D_1 \cap D_2 \cap D_3$ is non-empty.

Theorem 4.4. *Three discs, D_1, D_2 and D_3, have a non-empty intersection if and only if one of the following cases occur:*

(1) there is a disc whose center is in the other two discs; or

(2) the previous case does not occur and there exists a pair of discs for which one of both intersection points of their bordering circles lies in the remaining disc.

Proof. The *if*-direction is trivial. The *only if*-direction is less trivial. We will use the following abbreviations, $D = D_1 \cap D_2 \cap D_3$ and $C = \partial D$.

Assume D is non-empty and that neither (1) nor (2) holds. The intersection D is convex as it is the intersection of convex sets. We distinguish between the case where D is a point or and the case where D is not a point.

- Suppose D is the singleton $\{p\}$. This point p can not lie in the interior of the three discs, because D would not be a point then.

  Nor can p lie in the interior of two discs. If that would be the case then there exists a neighborhood of p that is part of the intersection of those two discs, say D_1 and D_2. Moreover p would be part of C_3 and this neighborhood would intersect the interior of D_3. This means D is not a point.

  So p must lie on the border of two discs, say D_1 and D_2, and p must also be part of D_3 because $D = \{p\}$. This contradicts our assumption that (2) does not hold.

- Assume D is not a point. All points on C belong to at least one C_i. If there is a point that does not belong to any C_i, then it is in the interior of all D_i and there exists a neighborhood of that point that is in the interior of all D_i and hence in D. That contradicts to the fact that this point is in C.

Furthermore, not all points of C belong to a single C_i. If that was the case then D_i would be part of (and equal to) D and its center would be inside the other two discs which contradicts the assumption that (1) does not hold.

So, C is made up of parts of the C_i, of which some may coincide but not all of them. When traveling along C we will encounter a point p that connects a part of a C_i and a part of a C_j, where $i \neq j$, that do not coincide, otherwise (1) must occur again which is a contradiction. However, this p also yields to a contradiction since it belongs to two different C_i, say C_1 and C_2, and is part of C hence D and D_3. This contradicts the assumption that (2) does not occur.

□

4.2.7.3 Translating the theorem in a formula

We can simplify the equations of the disks even further using coordinate transformations. By applying a translation, rotation and scaling we may assume that $(x_1, y_1) = (0, 0)$, $x_2 \geq 0$, $r_1 = 1$ and $y_2 = 0$. Using these simplifications and translating Theorem 4.4, we get the following formula.

$$\psi_1(x_2, r_2, x_3, y_3, r_3) := \left((-x_2)^2 \leq r_2^2 \ \wedge \ (-x_3)^2 + (-y_3)^2 \leq r_3^2\right) \ \vee$$
$$\exists x \exists y \left(x^2 + y^2 = 1 \ \wedge \ (x - x_2)^2 + y^2 = r_2^2 \ \wedge \ (x - x_3)^2 + (y - y_3)^2 \leq r_3^2\right).$$

This is a formula that decides if either the center of the first disc is part of the two other discs (see the first line), or if either there exists a point in the intersection of the first two circles that is part of the third disc (see the second line). All that remains now is making the expression

$$\exists x \exists y \left(x^2 + y^2 = 1 \ \wedge \ (x - x_2)^2 + y^2 = r_2^2 \ \wedge \ (x - x_3)^2 + (y - y_3)^2 \leq r_3^2\right)$$

quantifier free.

To do this we assume that C_1 and C_2 do not coincide but have a non-empty intersection. This is equivalent to $x_2 \neq 0 \ \wedge \ x_2 \leq r_2 + 1$. Next, we need to compute the point(s) where C_1 and C_2 intersect. These are given by

$$\begin{cases} x^2 + y^2 = 1 \\ (x - x_2)^2 + y^2 = r_2^2 \end{cases} \quad \text{or} \quad \begin{cases} x = \frac{x_2^2 + 1 - r_2^2}{2x_2} \\ y = \pm\sqrt{1 - \left(\frac{x_2^2 + 1 - r_2^2}{2x_2}\right)^2} \end{cases}$$

$$\text{or} \quad \begin{cases} x = \frac{x_2^2 + 1 - r_2^2}{2x_2} \\ y = \pm\sqrt{\left(1 - \frac{x_2^2 + 1 - r_2^2}{2x_2}\right)\left(1 + \frac{x_2^2 + 1 - r_2^2}{2x_2}\right)} \end{cases}$$

$$\text{or} \quad \begin{cases} x = \frac{x_2^2+1-r_2^2}{2x_2} \\ y = \pm\frac{1}{2x_2}\sqrt{\left(r_2^2-(x_2-1)^2\right)\left((1+x_2)^2-r_2^2\right)}. \end{cases}$$

If $y = 0$, then verifying if that single point of intersection is part of D_3 is easy, one only needs to verify whether

$$\left(\frac{x_2^2+1-r_2^2}{2x_2} - x_2\right)^2 + y_3^2 \leq r_3^2$$

or equivalently $\left(1-r_2^2-x_2^2\right)^2 + 4x_2^2y_3^2 \leq 4x_2^2r_3^2$. If $y \neq 0$, then verifying if one both points of intersection is part of D_3 is less trivial, since this involves square roots

$$\left(\frac{x_2^2+1-r_2^2}{2x_2} - x_3\right)^2 + \left(\pm\frac{1}{2x_2}\sqrt{\left(r_2^2-(x_2-1)^2\right)\left((1+x_2)^2-r_2^2\right)} - y_3\right)^2 \leq r_3^2$$

$$\text{or} \quad \left(x_2^2+1-r_2^2-2x_2x_3\right)^2 + \left(\pm\sqrt{\left(r_2^2-(x_2-1)^2\right)\left((1+x_2)^2-r_2^2\right)} - 2x_2y_3\right)^2 \leq 4x_2^2r_3^2$$

$$\text{or} \quad \left(x_2^2+1-r_2^2-2x_2x_3\right)^2 + \left(r_2^2-(x_2-1)^2\right)\left((1+x_2)^2-r_2^2\right) + (2x_2y_3)^2 \pm 4x_2y_3\sqrt{\left(r_2^2-(x_2-1)^2\right)\left((1+x_2)^2-r_2^2\right)} \leq 4x_2^2r_3^2$$

$$\text{or} \quad \left(x_2^2+1-r_2^2-2x_2x_3\right)^2 + \left(r_2^2-(x_2-1)^2\right)\left((1+x_2)^2-r_2^2\right) + (2x_2y_3)^2 - 4x_2^2r_3^2 \leq \pm 4x_2y_3\sqrt{\left(r_2^2-(x_2-1)^2\right)\left((1+x_2)^2-r_2^2\right)}.$$

This is almost a $\mathsf{FO}(+,\times,<,0,1)$-formula except for the square root. However, the square root can be eliminated as we will show next. The previous expression is of the form $L \leq \pm a\sqrt{W}$. The presence of the $\pm$ simplifies this a lot, this means either sign of the square root will do, and also that we may assume the right hand-side is positive. Of course the square root must exist as well, this means $W \geq 0$.

This expression can then be simplified to

$$W \geq 0 \ \wedge \ \left(L \leq 0 \ \vee \ L^2 \leq a^2 W\right)$$

and gives us the expression

$$\begin{aligned}
&\Phi_2(x_2, r_2, x_3, y_3, r_3) := \\
&\quad \left(r_2^2 - (x_2-1)^2\right)\left((1+x_2)^2 - r_2^2\right) \geq 0 \wedge \Bigg(\left(x_2^2 + 1 - r_2^2 - 2x_2x_3\right)^2 \\
&\qquad + \left(r_2^2 - (x_2-1)^2\right)\left((1+x_2)^2 - r_2^2\right) + (2x_2y_3)^2 - 4x_2^2r_3^2 \leq 0 \\
&\quad \vee \ \Bigg(\left(x_2^2 + 1 - r_2^2 - 2x_2x_3\right)^2 + \left(r_2^2 - (x_2-1)^2\right)\left((1+x_2)^2 - r_2^2\right) \\
&\quad + (2x_2y_3)^2 - 4x_2^2r_3^2\Bigg)^2 \leq (4x_2y_3)^2\left(r_2^2 - (x_2-1)^2\right)\left((1+x_2)^2 - r_2^2\right)\Bigg).
\end{aligned}$$

4.2.7.4 The safety formula

Now, all that remains to be constructed is a formula that returns the convenient coordinates and a formula that guarantees that C_1 and C_2 actually intersect for safety, i.e., to exclude the case of empty intersection. The latter is constructed as follows. The formula $\varphi(x_1, y_1, r_1, x_2, y_2, r_2)$ returns *true* if and only if the two circles, with centers (x_1, y_1) and (x_2, y_2) and radii r_1 and r_2 respectively, have a distance between their centers that is not larger that the sum of their radii and not equal to zero to ensure they do not coincide. We have

$$\varphi(x_1, y_1, r_1, x_2, y_2, r_2) := 0 < (x_2 - x_1)^2 + (y_2 - y_1)^2 \leq (r_1 + r_2)^2 \ .$$

The formula $\phi(x_1, y_1, r_1, x_2, y_2, r_2)$ returns TRUE if and only if the second circle is not fully enclosed by the first, i.e., the sum of the distance between the centers plus the second radius is bigger than the first radius and vice versa. We can write

$$\phi(x_1, y_1, r_1, x_2, y_2, r_2) := (x_2 - x_1)^2 + (y_2 - y_1)^2 \geq (r_1 - r_2)^2 \ .$$

These two safety conditions give us our safety formula

$$\begin{aligned}
&\Phi_{\text{safe}}(x_1, y_1, r_1, x_2, y_2, r_2) := \\
&\qquad\qquad \varphi(x_1, y_1, r_1, x_2, y_2, r_2) \ \wedge \ \phi(x_1, y_1, r_1, x_2, y_2, r_2).
\end{aligned}$$

4.2.7.5 The change of coordinates

The transformation consists of a translation, rotation and scaling. The translation moves the first circle's center to the origin. The rotation aligns the second centre with the x-axis. Finally the scaling ensures that the first circle's radius is equal to one. First, the translation $T(x, y) := (x - x_1, y - y_1)$. The rotation is

$$R(x,y) := \frac{1}{\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}} \begin{pmatrix} x_2-x_1 & y_2-y_1 \\ y_1-y_2 & x_2-x_1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

and finally, the scaling is $S(x, y) := \frac{1}{r_1}(x, y)$. The transformation is then a composition of those three transformations $A(x, y) = (S \circ R \circ T)(x, y) :=$

$$\left(\frac{(x_2-x_1)(x-x_1)+(y_2-y_1)(y-y_1)}{r_1\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}}, \right.$$
$$\left. \frac{(y_1-y_2)(x-x_1)+(x_2-x_1)(y-y_1)}{r_1\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}} \right).$$

The following formula takes three circles with centers (x_i, y_i) and radii r_i respectively, and transforms them in three new circles where the first circle has center $(0, 0)$ and radius 1, the second circle has center $(x'_2, 0)$ and radius r'_2 and the third circle has center (x'_3, y'_3) and radius r'_3.

$$\begin{aligned} \Phi_{\text{transformation}}(x_1,y_1,r_1,x_2,y_2,r_2,x_3,y_3,r_3,x'_2,r'_2,x'_3,y'_3,r'_3) := \\ x'_2 = \frac{\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}}{r_1} \ \wedge \ r'_2 = \frac{r_2}{r_1} \ \wedge \ r'_3 = \frac{r_3}{r_1} \\ \wedge \ x'_3 = \frac{(x_2-x_1)(x_3-x_1)+(y_2-y_1)(y_3-y_1)}{r_1\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}} \\ \wedge \ y'_3 = \frac{(y_1-y_2)(x_3-x_1)+(x_2-x_1)(y_3-y_1)}{r_1\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}}. \end{aligned}$$

Note that this is not a $\mathsf{FO}(+, \times, <, 0, 1)$-formula anymore due to the square roots and fractions. This "formula" is meant to act like a function, which substitutes coordinates. The substituted coordinates have fractions and square roots but these can easily be disposed of when having the entire inequality on a common denominator, isolating the square root and squaring the inequality, as we showed in Section 4.2.7.3.

4.2.7.6 The formula for the alibi query at a fixed moment in time

First, we construct a formula that checks for any of two circles out of three if any of the conditions in Theorem 4.4 are satisfied.

$$\begin{aligned}
&\psi_{2/3}(x_1, y_1, r_1, x_2, y_2, r_2, x_3, y_3, r_3) \ := \exists x_2' \exists r_2' \exists x_3' \exists y_3' \exists r_3' \\
&\quad \big(\Phi_{\text{transformation}}(x_1, y_1, r_1, x_2, y_2, r_2, x_3, y_3, r_3, x_2', r_2', x_3', y_3', r_3') \wedge \\
&\big(\psi_1(x_2', r_2', x_3', y_3', r_3') \ \vee \ \Phi_{\text{safe}}(x_1, y_1, r_1, x_2, y_2, r_2) \ \wedge \ \Phi_2(x_2', r_2', x_3', y_3', r_3')\big) \\
&\quad \vee \ \Phi_{\text{transformation}}(x_1, y_1, r_1, x_3, y_3, r_3, x_2, y_2, r_2, x_2', r_2', x_3', y_3', r_3') \ \wedge \\
&\big(\psi_1(x_2', r_2', x_3', y_3', r_3') \ \vee \ \Phi_{\text{safe}}(x_1, y_1, r_1, x_3, y_3, r_3) \ \wedge \ \Phi_2(x_2', r_2', x_3', y_3', r_3')\big) \\
&\quad \vee \ \Phi_{\text{transformation}}(x_2, y_2, r_2, x_3, y_3, r_3, x_1, y_1, r_1, x_2', r_2', x_3', y_3', r_3') \ \wedge \\
&\big(\psi_1(x_2', r_2', x_3', y_3', r_3') \ \vee \ \Phi_{\text{safe}}(x_2, y_2, r_2, x_3, y_3, r_3) \ \wedge \ \Phi_2(x_2', r_2', x_3', y_3', r_3')\big)\big) \, .
\end{aligned}$$

This formula is all we need to incorporate Helly's theorem in our final formula. Four discs have a non-empty intersection if and only if the following formula is satisfied

$$\begin{aligned}
\psi_{\text{alibi}}(x_1, y_1, r_1, x_2, y_2, r_2, x_3, y_3, r_3, x_4, y_4, r_4) := \\
&\psi_{2/3}(x_1, y_1, r_1, x_2, y_2, r_2, x_3, y_3, r_3) \ \wedge \\
&\psi_{2/3}(x_1, y_1, r_1, x_2, y_2, r_2, x_4, y_4, r_4) \ \wedge \\
&\psi_{2/3}(x_1, y_1, r_1, x_3, y_3, r_3, x_4, y_4, r_4) \ \wedge \\
&\psi_{2/3}(x_2, y_2, r_2, x_3, y_3, r_3, x_4, y_4, r_4).
\end{aligned}$$

This is almost a quantifier free-formula except for the fractions and square roots. However, as we showed before these can easily be disposed of. We omitted these tedious conversions for the sake of clarity.

4.2.7.7 Notes on experiments

First, we remark that the formula above is a lot shorter than the formula ψ_{alibi}, so it is reasonable to assume that this formula evaluates even faster than ψ_{alibi}. Before we started looking for a $\mathsf{FO}(+, \times, <, 0, 1)$-formula to answer the alibi query, we tried several software packages, that support quantifier-elimination, to produce a formula for us. In both cases, the general alibi query and the alibi query for a fixed moment in time, the quantifier-elimination methods failed to produce an answer at all after prolonged sessions. This was expected and, at the same time, a motivation to find such a formula ourselves.

For the non-parametric case however, MATHEMATICA returns an answer in a matter of hundredths of a second. In this case, the non-parametric alibi query for a fixed moment in time, the general methods do produce an answer

efficiently. The relative differences were inconsistent, the same parameters would take, for example, .014 and .031 seconds, depending on the other processes the system was running and which we could not control. For this reason we decided not to include experimental results in this case.

4.3 Conclusion

In this chapter, we proposed a method that decides if two space-time prisms have a non-empty intersection or not. Existing quantifier-elimination methods can achieve this already through means of quantifier elimination though not in a reasonable amount of time. Deciding intersection of concrete space-time prisms takes of the order of minutes, while the parametric case can be measured at least in days if a solution would ever be obtained. The parametric solution we laid out in this paper only takes a few milliseconds or less.

The solution we present is almost a first-order formula containing square root-expressions. These can easily be disposed of using repeated squarings and adding extra conditions, thus obtaining a true quantifier free-expression for the alibi query.

We also give a solution to the alibi query at a fixed moment in time.

The solutions we propose are based on geometric argumentation and they illustrate the fact that some practical problems require creative solutions, where at least in theory, existing systems could provide a solution.

5

Space-time prisms on road networks

Until now, space-time prisms have been studied for unconstrained movement in the two-dimensional plane. In this chapter, we study space-time prims for objects that are constraint to travel on a road network. Movement on a road network can be viewed as essentially one-dimensional. We describe the geometry of a space-time prism and give an algorithm to compute and visualize space-time prisms on a road network. For experiments and illustration, we have implemented this algorithm in MATHEMATICA.

Furthermore, we study the *alibi query*, which ask whether two moving objects could have possibly met or not. This comes down to deciding if the chains of space-time prisms produced by these moving objects intersect. We give an efficient algorithm to answer the alibi query for moving objects on a road network. This algorithm also determines where and when two moving objects may have met.

This chapter is organized as follows. We start with the defining space-time prisms on road networks. In Section 5.2, we first give a theorem that is the heart of the algorithm we develop. In the same section, we give the algorithms to construct and visualize space-time prisms and their spatial projections on road networks. In Section 5.3, we give a solution to the alibi query for objects moving in one dimensional-space and then for objects moving in a road network.

5.1 Space-time prisms on road networks

The next step is to determine these space-time prisms on a road network such that they can easily be queried and visualized afterwards.

This does not simply amount to taking the intersection of a space-time prism representing movement in $\mathbf{R}^2$ and the road network. To see this, consider the projection of the unconstrained space-time prism along the time axis onto the (x, y)-plane. This projection is an ellipse with foci the points of departure and arrival, i.e., $p_i = (\mathsf{x}_i, \mathsf{y}_i)$ and $p_{i+1} = (\mathsf{x}_{i+1}, \mathsf{y}_{i+1})$. We recall that at a time t, $\mathsf{t}_i \leq t \leq \mathsf{t}_{i+1}$, the object's distance to p is at most $\mathsf{v}_i(t - \mathsf{t}_i)$ and its distance to q is at most $\mathsf{v}_i(\mathsf{t}_{i+1} - t)$. Adding those distances gives $\mathsf{v}_i(t - \mathsf{t}_i) + \mathsf{v}_i(\mathsf{t}_{i+1} - t)\mathsf{v}_i(\mathsf{t}_{i+1} - \mathsf{t}_i)$, which is constant. Therefor all possible points a moving object with speed limit $v_{\max}$ could have visited must lie within this ellipse with foci p_i and p_{i+1}. Moreover, the sum of their distances to p_i and p is less or equal to $\mathsf{v}_i(\mathsf{t}_{i+1} - \mathsf{t}_i)$. In this ellipse, any trajectory that touches the border of the ellipse and has more than two straight line segments is longer than $\mathsf{v}_i(\mathsf{t}_{i+1} - \mathsf{t}_i)$. This particular trajectory lies in the ellipse of the unconstrained space-time prism and the road network, but a space-time prism on the road network does not capture it because there are points on it which can be reached in time, but from which the destination cannot be reached in time and vice versa. Figure 5.1 depicts such a situation. There is no path on the

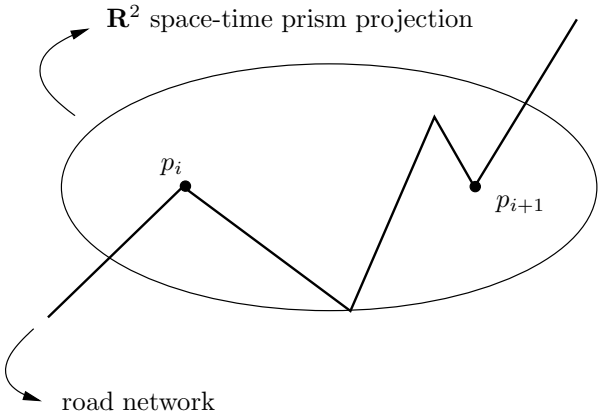


Figure 5.1: Road network space-time prisms can not be easily derived from space-time prisms in $\mathbf{R}^2$.

road network from p that reaches q in the given time interval. The intersection of the space-time prism with the road network is nonempty, whereas the road network space-time prism clearly is.

To define space-time prisms on a road network, we need to define an ap-

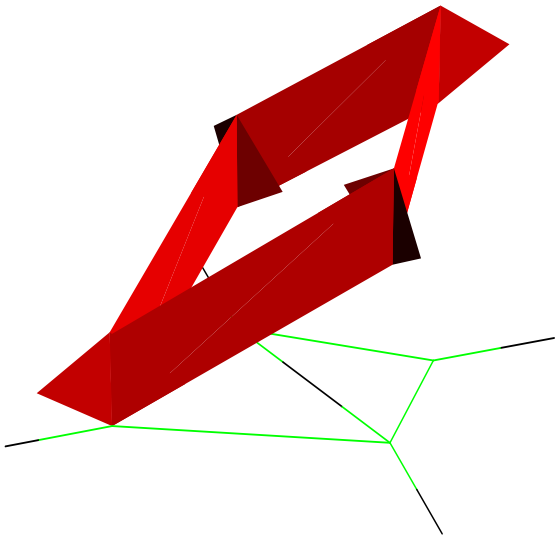


Figure 5.2: Space-time prism (red) on road networks (green and black) and its spatial projection (green).

propriate distance function on the road network. The distance measure that we use is derived from the *shortest path*-distance used in graph theory [3].

Suppose we have road network RN, given by its set of vertices V and set of labeled edges E. Let $p = (x_p, y_p)$ and $q = (x_q, y_q)$ be two points on that road network RN. The points p and q are not necessarily vertices. Suppose p lies on (the embedding of) the edge $((x_{p,0}, y_{p,0}), (x_{p,1}, y_{p,1}))$ and q lies on the edge $((x_{q,0}, y_{q,0}), (x_{q,1}, y_{q,1}))$. We construct a new road network RN_{pq} from RN. Its set of vertices is $\mathsf{V_{pq}} = \mathsf{V} \cup \{p, q\}$ and its set of edges is $\mathsf{E_{pq}} = \mathsf{E} \cup \{((x_{p,0}, y_{p,0}), (x_p, y_p)), ((x_p, y_p), (x_{p,1}, y_{p,1})), ((x_{q,0}, y_{q,0}), (x_q, y_q)), ((x_q, y_q), (x_{q,1}, y_{q,1}))\}$. So, we have split the edges on which p and q are located in p and q. For what concerns the labels, we keep the speed limits of the original edges for the split edges and the time spans of the new edges are computed according to Definition 2.10.

It is precisely this construction we need to define the distance along the road network RN and space-time prism on RN.

Definition 5.1. Let RN be a road network, $p, q \in \mathsf{RN}$ and let V_{pq} be the set of vertices $\mathsf{V} \cup \{p, q\}$ and let E_{pq} equal the set of edges obtained from E by adding p and q to the vertex set. The *road network time* between p and q, denoted by $\mathsf{d}_{\mathsf{RN}}(p, q)$, is the shortest-path distance[1] between p and q in the graph $(\mathsf{V}_{pq}, \mathsf{E}_{pq})$, with respect to the time-span labelling of the edges.

A path, i.e., a list of adjacent edges that connect p and q, from p to q whose length, with respect to the time-span labelling of the edges, equals $\mathsf{d}_{\mathsf{RN}}(p, q)$ is called a *fastest path from p to q*. □

We note that the *road network time* between p and q in the above definition has minimal total weight and returns the earliest possible time in which you can reach p from q and vice versa. The metric that we describe takes two points from a road network and returns the shortest time needed to get from one to the other when travelling at the allowed maximal speed at each segment.

We remark that if all edges in road network have the same speed limit $v_{\max}$, then the metric defined in Definition 5.1 is the shortest-path metric (up to a scaling factor $v_{\max}$) on the graph embedding RN. If, on the other hand, different speed limits appear per edge, then the shortest paths are not always the fastest paths. In Figure 5.9, we can see that neither one of the two shortest paths in the road network is also the fastest path. The fastest path starts at the left-most node, goes to the upper node and then the lower node and ends at the right-most node.

A space-time prism on a road network is the geometric location in $\mathbf{R} \times \mathsf{RN} \subset \mathbf{R} \times \mathbf{R}^2$ of all points a moving object could have visited when travelling, restricted to RN, from an origin p to a destination q with in a time-frame ranging from t_p to t_q, respecting the speed limits on the edges of RN. We define this more formally. Given a road network RN, points p, q and u on RN, and time moments t_p and t_q for p and q, we write t_u^- to abbreviate $t_p + \mathsf{d}_{\mathsf{RN}}(p, u)$ and t_u^+ to abbreviate $t_q - \mathsf{d}_{\mathsf{RN}}(u, q)$. We note that in the notationt t_u^- and t_u^+, p and q are absent although these values are dependent on them, we choose not to overload the notation in this case because it is usually clear from the context what p and q are.

Definition 5.2. Let RN be a road network, let $p, q \in \mathsf{RN}$. The *space-time prism on the road network* between (t_p, p) and (t_q, q), with respect to the speed limits of RN is denoted by $\mathcal{P}^{\mathsf{RN}}(t_p, p, t_q, q)$ and is defined as the set of (t, x, y) tuples for which

$$\begin{cases} u = (x, y) \in \mathsf{RN}, \\ \mathsf{d}_{\mathsf{RN}}(p, u) + \mathsf{d}_{\mathsf{RN}}(u, q) \leq (t_q - t_p), \\ t_u^- \leq t \leq t_u^+. \end{cases}$$

[1] We mean the single-source shortest-path distance that is commonly used in graph theory and that can be computed efficiently by Dijkstra's algorithm [3].

The *space-time prism on the road network* between (t_p, p) and (t_q, q), with respect to a general maximal speed $v_{\max}$ is the space-time prism on RN after relabelling all edges of RN with speed limit $v_{\max}$. □

5.2 Properties of space-time prisms on road networks

In the main result of this section, we describe the structure of a space-time prism. This composition is turned into an algorithm to compute and visualize a space-time prism and its spatial projection, later on. In Section 5.3, we use the techniques provided by this theorem, to answer the alibi query for road networks.

Our inspiration for the following theorem comes from the observation that a one-dimensional space-time prism can be dissected as pictured in Figure 5.3. The two diamonds represent all trajectories from a node to itself given bounds on time and speed. The parallelogram on the right represents all travel from one node to an adjacent one given the time limits and the ruling speed limit on that segment. The idea is that if we know the time limits in each node, and the ruling speed limit on the edge connecting the nodes, we can reconstruct the entire prism one edge at a time.

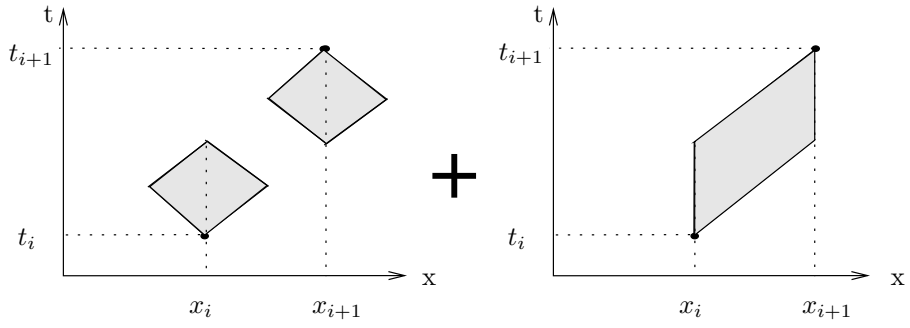


Figure 5.3: A dissection of a one-dimensional space-time prism.

Theorem 5.3. *Let* RN *be a road network given by the vertex set* V *and the edge set* E *and network time span labels for the edges. Let* p *and* q *be two vertices on* RN.

1. *A vertex* u *is part of* $\mathcal{P}^{\mathsf{RN}}(t_p, p, t_q, q)$ *if and only if* $\mathsf{d}_{\mathsf{RN}}(p, u) + \mathsf{d}_{\mathsf{RN}}(u, q) \leq (t_q - t_p)$;

2. *If the vertex u is in $\mathcal{P}^{\mathsf{RN}}(t_p, p, t_q, q)$, then $\mathcal{P}^{\mathsf{RN}}(t_u^-, u, t_u^+, u) \subseteq \mathcal{P}^{\mathsf{RN}}(t_p, p, t_q, q)$;*

3. *If the vertices u and w are in the space-time prism $\mathcal{P}^{\mathsf{RN}}(t_p, p, t_q, q)$, then $\mathcal{P}^{\mathsf{RN}}(t_u^-, u, t_w^+, w) \subseteq \mathcal{P}^{\mathsf{RN}}(t_p, p, t_q, q)$;*

4. *Moreover, the space-time prism $\mathcal{P}^{\mathsf{RN}}(t_p, p, t_q, q)$ equals*

$$\bigcup_{\substack{u,\, v \,\in\, \mathsf{V},\\ \mathsf{d}_{\mathsf{RN}}(p,\,u) + \mathsf{d}_{\mathsf{RN}}(u,\,q) \,\leq\, (t_q - t_p)\\ \mathsf{d}_{\mathsf{RN}}(p,\,w) + \mathsf{d}_{\mathsf{RN}}(w,\,q) \,\leq\, (t_q - t_p)\\ t_p + \mathsf{d}_{\mathsf{RN}}(p,\,u) \,<\, t_q - \mathsf{d}_{\mathsf{RN}}(w,\,q)}} \mathcal{P}^{\mathsf{RN}}(t_p + \mathsf{d}_{\mathsf{RN}}(p,u), u, t_q - \mathsf{d}_{\mathsf{RN}}(w,q), w).$$

Proof of Theorem 5.3. Part (i) is trivial. For Part (ii) and Part (iii), it suffices to remark that an object travelling past a vertex u can not arrive there sooner than $t_u^- = t_p + \mathsf{d}_{\mathsf{RN}}(p,u)$ and can not leave the vertex u later than $t_u^+ = t_q - \mathsf{d}_{\mathsf{RN}}(u,q)$. In Part (ii), $\mathcal{P}^{\mathsf{RN}}(t_u^-, u, t_u^+, u)$ consists of all points that are reachable from u and from which one can return to u in the remaining time interval $[t_u^-, t_u^+]$, as illustrated in Figure 5.4. In Part (iii) is described which vertices can be reached in the remaining time, as illustrated in Figure 5.5.

From Parts (i), (ii) and (iii) follows the inclusion $\supseteq$ of Part (iv). The other inclusion is clear: $\mathcal{P}^{\mathsf{RN}}(t_p, p, t_q, q)$ contains all vertices u and w that can be reached in time and all points that can be reached from these vertices in the remaining time.

□

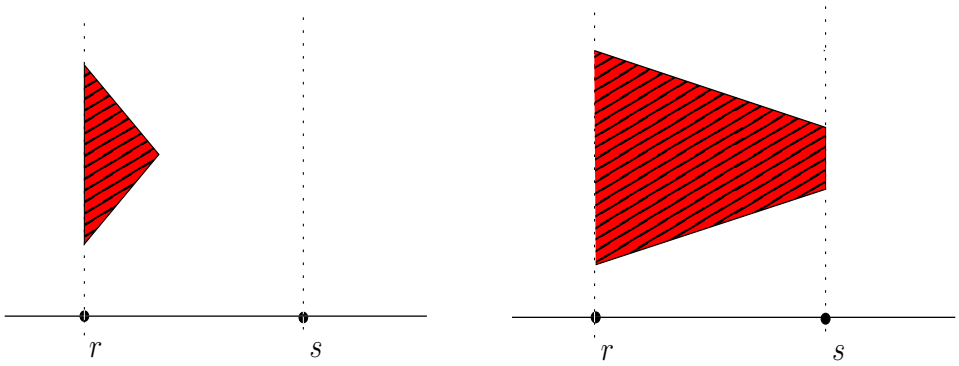


Figure 5.4: Cases 1 and 2.

We remark that it is sufficient in Part (iii) of Theorem 5.3, to consider adjacent vertices u and w. So, Part (iv) of Theorem 5.3 states that it is

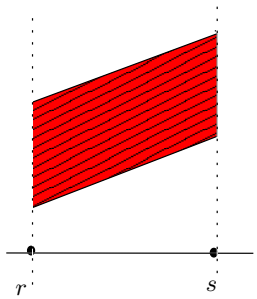


Figure 5.5: Case 3.

sufficient to compute space-time prisms starting and ending in a vertex and between adjacent vertices in the road network. The union of all these make up the entire space-time prism.

5.2.1 Computing and visualizing space-time prisms on road networks

In this section, we describe an algorithm to compute a space-time prism on a road network. The direct application of this computation is the visualization of the space-time prism. Further on, we also use this computation in querying, more specifically in computing, answering and visualising the alibi query on road networks.

Theorem 5.3 is the key to a relatively simple algorithm to compute the space-time prisms. The algorithm to compute (visualise) road network space-time prisms can be split into two major parts. A pre-computation and the actual computation and drawing.

The input is a road network RN in which the edges are labeled with speed limits. Also an origin p and destination q on RN are given and a departure time t_p and arrival time t_q. Let $v_{\max}$ denote the maximum of all speed limits occurring in RN.

The pre-computation can then be split into two parts and reused for both 2-dimensional and 3-dimensional visualization. The algorithm takes as input a road network RN given as a set of vertices, edges and weights, a starting point and time p and t_p, and an arrival point and time q and t_q. It outputs a modified road network RN', a set of time pairs $\{(t_u^-, t_u^+)\} \mid u \in \mathsf{RN}'\}$ denoting the bounds of the intervals in time of the space-time prism $\mathcal{P}^{\mathsf{RN}}(t_p, p, t_q, q)$ at each vertex, a subset of vertices $\mathsf{V}_{\mathcal{P}}$ of V that are part of the space-time prism and a subset of edges $\mathsf{E}_{\mathcal{P}}$ of E that are part of or intersect the space-time prism.

PRECOMP: input= $(\mathsf{RN}, p, t_p, q, t_q)$;
output= $(\mathsf{RN}', \{(t_u^-, t_u^+)\} \mid u \in \mathsf{RN}'\}, \mathsf{V}_\mathcal{P}, \mathsf{E}_\mathcal{P})$

Step 1. First, we add the vertices p and q to the vertex set of RN. Then we isolate that part of RN that can actually be part of the spatial projection of the space-time prism. To do this we restrict our attention to the smallest part of RN that is in $\pi_{xy}\mathcal{P}(t_p, p, t_q, q, v_{\max})$ where $\mathcal{P}(t_p, p, t_q, q, v_{\max})$ is the unconstrained space-time prism for movement in $\mathbf{R}^2$, π_{xy} is an operator that projects a set in $\mathbf{R} \times \mathbf{R}^2$ onto the (x, y)-plane and $v_{\max}$ is the maximum of all weights in RN. The set $\pi_{xy}\left(\mathcal{P}(t_p, p, t_q, q, v_{\max})\right)$ is a set bounded by an ellipse, see Section 2.3. First, we select all the edges that have at least one vertex that is inside this projected set and select the vertices that bound those edges. So, we obtain a new reduced road network RN' consisting of all edges that intersect $\pi_{xy}\left(\mathcal{P}^{\mathbf{R}^2}(t_p, p, t_q, q, v_{\max})\right)$.

Step 2. Now we apply a single-source shortest path algorithm (e.g. Dijkstra's algorithm) twice on the graph RN'. Once to compute all the road network times spans to p, and once more to compute all the road network time spans to q.

For each vertex u in RN' we store its road network time to p, i.e., the arrival time $t_u^- = t_p + \mathsf{d}_{\mathsf{RN}}(p, u)$, its road network time to q, i.e., the time to leave $t_u^+ = t_q - \mathsf{d}_{\mathsf{RN}}(u, q)$. We also store the vertices u for which $\mathsf{d}_{\mathsf{RN}}(p, u) + \mathsf{d}_{\mathsf{RN}}(u, q) \leq (t_q - t_p)$ in the set $\mathsf{V}_\mathcal{P}$. In the set $\mathsf{E}_\mathcal{P}$ we store all edges that connect to at least one vertex in $\mathsf{V}_\mathcal{P}$.

With the information of the the pre-computation step, we can easily visualize a space-time prism above the road-network. We look at two problems. Firstly, we determine or visualise the spatial projection of the space-time prism and later, we determine or visualize a space-time prism in time-space on top of the projection.

To visualize the spatial projection, i.e., to color all the possible places in a road network that could have been visited when travelling from p to q with a speed that never exceeded the local speed limit, we give the following algorithm.

2D-SPACE-TIME PRISM: input= $(\mathsf{RN}', \{(t_u^-, t_u^+\} \mid u \in \mathsf{RN}'\}, \mathsf{V}_\mathcal{P}, \mathsf{E}_\mathcal{P})$;
output= drawing of $\pi_{xy}\left(\mathcal{P}^{\mathsf{RN}}(t_p, p, t_q, q,)\right)$.

For every edge (r, s) in $\mathsf{E}_\mathcal{P}$, with $r = (x_r, y_r)$ and $s = (x_s, y_s)$, in $\mathsf{E}_\mathcal{P}$ let its speed limit be v_{rs} and time span be w_{rs} and set $d_{rs} = \sqrt{(x_s - x_r)^2 + (y_s - y_r)^2} = v_{rs}w_{rs}$.

- If $t_r^- + w_{rs} \leq t_s^+$ or $t_s^- + w_{rs} \leq t_r^+$ then color the entire line segment (x_r, y_r, x_s, y_s) and exit;
- else if $r \in \mathsf{V}_\mathcal{P}$ and $t_r^- + w_{rs} > t_s^+$ then color the line segment with endpoints (x_r, y_r, x_0, y_0), where $(x_0, y_0) = (x_r, y_r) + \frac{(t_r^+ - t_r^-)}{2} v_{rs} \frac{(x_s - x_r, y_s - y_r)}{d_{rs}}$; and
- if $s \in \mathsf{V}_\mathcal{P}$ and $t_s^- + w_{rs} > t_r^+$ then color the line segment with endpoints (x_0, y_0, x_s, y_s), where $(x_0, y_0) = (x_s, y_s) + \frac{(t_s^+ - t_s^-)}{2} v_{rs} \frac{(x_r - x_s, y_r - y_s)}{d_{rs}}$.

The equation $(x_0, y_0) = (x_r, y_r) + \frac{(t_r^+ - t_r^-)}{2} v_{rs} \frac{(x_s - x_r, y_s - y_r)}{d_{rs}}$ needs more explanation. In the second step of the above algorithm, we are coloring the projection of Case 1 of Figure5.4. The point (x_0, y_0) is the projection of the right most vertex of the triangle in Figure 5.4. Its coordinates are those of the vertex r plus a unit vector pointing from r to s, $\frac{(x_s - x_r, y_s - y_r)}{d_{rs}}$, times the distance of that vertex from r, which equals half the time that can be spent at r, $\frac{(t_r^+ - t_r^-)}{2}$, times the speed limit on the edge between r and s, v_{rs}. To visualize the space-time prism in time-space, we also iterate over all edges, but instead of coloring 2-dimensional line segments we have to color one or two 3-dimensional polygons per edge. We use the following notation for polygons, $\langle p_1, \ldots, p_k \rangle$ is the polygon obtained by connecting consecutive points p_i with p_{i+1} in this list and connecting the first point p_1 with the last point p_k in the list.

3D-SPACE-TIME PRISM: input= $(\mathsf{RN}', \{(t_u^-, t_u^+\} \mid u \in \mathsf{RN}'\}, \mathsf{V}_\mathcal{P}, \mathsf{E}_\mathcal{P})$;
output= drawing of $\mathcal{P}^{\mathsf{RN}}(t_p, p, t_q, q,)$.

For each edge (r, s) in $\mathsf{E}_\mathcal{P}$, with $r = (x_r, y_r)$ and $s = (x_s, y_s)$, we do the following for r and s. There are several possible cases one needs to consider and they are illustrated in Figure 5.4, corresponding to Part (2) of Theorem 5.3 and Figure 5.5 corresponding to Part (3) of Theorem 5.3.

- These are Cases 1 and 2, illustrated in Figure 5.4: If $w_{rs} \leq \frac{(t_r^+ - t_r^-)}{2}$, then draw the polygon $\langle (t_0, x_0, y_0), (t_r^+, x_r, y_r), (t_r^-, x_r, y_r), (t_0, x_0, y_0) \rangle$ where $t_0 = \frac{t_r^- + t_r^+}{2}$ and

$$(x_0, y_0) = (x_r, y_r) + \frac{(t_r^+ - t_r^-)}{2} v_{rs} \frac{(x_s - x_r, y_s - y_r)}{d_{rs}}.$$

  Otherwise, draw the polygon $\langle (t_1, x_s, y_s), (t_r^+, x_r, y_r), (t_r^-, x_r, y_r), (t_0, x_s, y_s), (t_1, x_s, y_s) \rangle$ where $t_0 = t_r^- + w_{rs}$ and $t_1 = t_r^+ - w_{rs}$;

- This is Cases 3 illustrated in Figure 5.5: In this case one is able to reach s from r before one has to leave s again. In other words, $t_s^+ \geq t_r^- + w_{rs}$. In which case the polygon $\langle (t_r^-, x_r, y_r), (t_1, x_r, y_r), (t_s^+, x_s, y_s), (t_0, x_s, y_s) \rangle$ is drawn where $t_0 = t_r^- + w_{rs}$ and $t_1 = t_s^+ - w_{rs}$;
- Repeat the previous steps with the indices r and s interchanged.

The correctness of the algorithms follows from Theorem 5.3. The end result on each segment is either a polygon as drawn in Figure 5.4, Figure 5.5 or a combination thereof as illustrated in Figure 5.6.

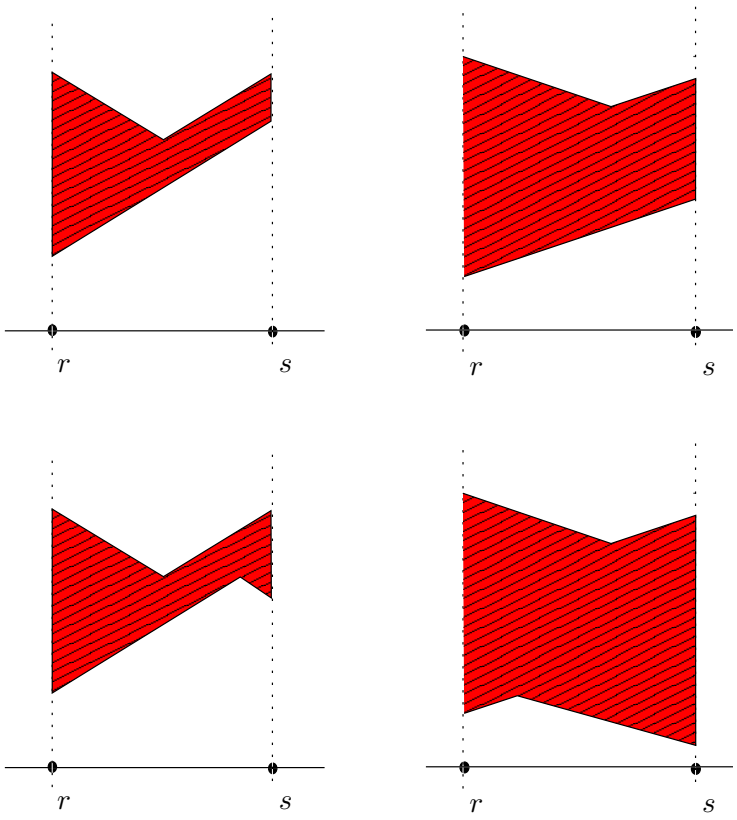


Figure 5.6: A combination of Figures 5.4 and 5.5.

5.2.2 Implementation and illustration

As proof of concept, we implemented algorithms 2D-SPACE-TIME PRISM and 3D-SPACE-TIME PRISM of Section 5.2.1 in Wolfram's MATHEMATICA [32]. This

implementation is available at [24]. In this section, we illustrate its working. Figure 5.8 gives three examples of space-time prisms (red) and their spatial projection (green) above a road network (black and green).

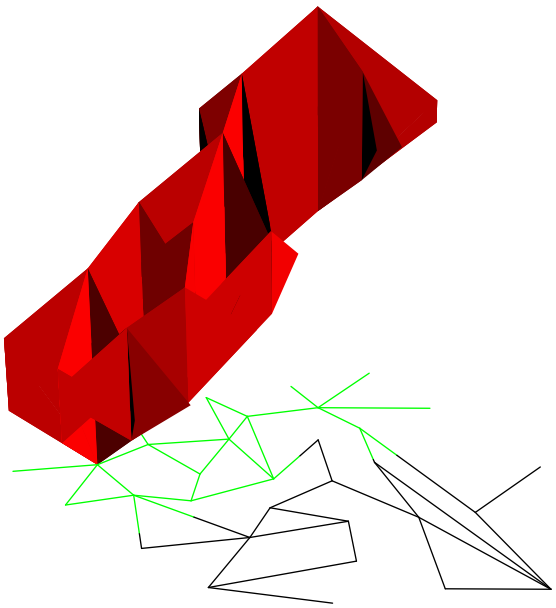


Figure 5.7: A space-time prism (red) on a road network (green and black) and its spatial projection (green).

5.2.3 Complexity considerations

Let the number of vertices in the road network be n. Considering from a practical point of view, we may also say the number of edges in the network is $\mathcal{O}(n)$. This comes from assuming that every vertex, in a real-life road network, connects to rarely more than four edges.

Selecting the vertices that are within the unconstrained projected space-time prism, i.e., within the ellipse, can be done by going over all vertices and selecting those that satisfy the inequality that describes the area bounded by the ellipse. Starting out with ordered vertices according to their coordinates can significantly reduce this complexity. However, in the worst case almost all

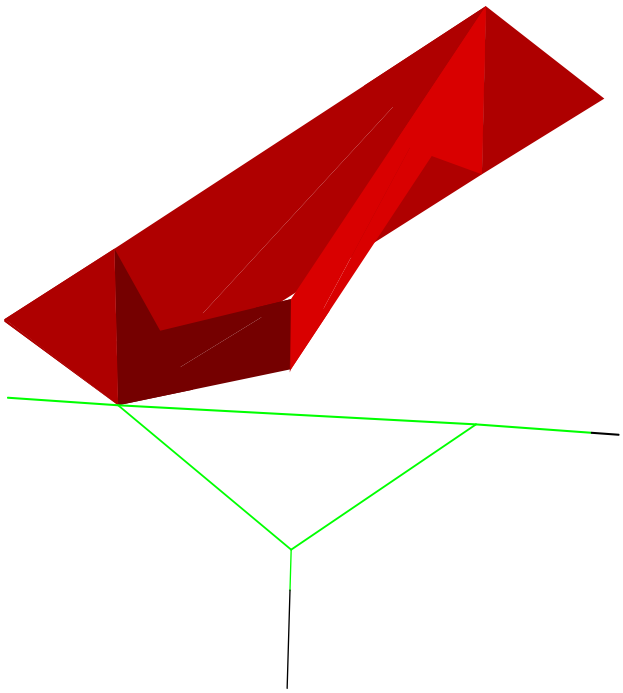


Figure 5.8: A space-time prism (red) on a road network (green and black) and its spatial projection (green).

vertices will be selected, which is something we can not *a priori* estimate.

This selection takes $\mathcal{O}(n)$ steps and we assume the number of selected vertices is again $\mathcal{O}(n)$. This is an overestimation in large road-networks, especially when sample points succeed each other closely in time. Assuming that the vertices are ordered, we can say that the selection step takes $\mathcal{O}(m)$ steps where m is the number of vertices selected within the ellipse and $m \ll n$.

The next step is running Dijkstra's algorithm, this takes $\mathcal{O}(m^2)$ time and is the bottleneck of our algorithm.

The rest, drawing in dimension two or three, only takes at most a fixed number of operations per edge. In the 2D case we draw at most two line-segments per edge, and in three dimensions at most four polygons. We also process every edge only once. This step takes again $\mathcal{O}(m)$ time.

From this we can conclude that visualizing space-time prisms takes $\mathcal{O}(m^2)$ time, with $m \ll n$.

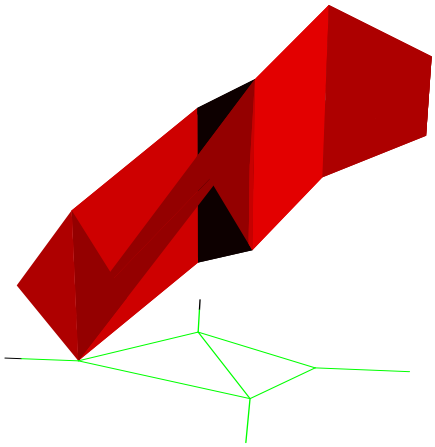


Figure 5.9: Road Network space-time prism where the fastest path does not coincide with the shortest.

5.3 The Alibi Query

5.3.1 The alibi query on road networks

Deciding whether two space-time prisms on a road network have an empty intersection is not too difficult once we have a way to visualize space-time prisms. We can proceed as follows. We first select the vertices that appear in both space-time prisms and collect the edges that have at least one vertex in that intersection. Then we iterate over all these edges and perform intersection tests on the polygons that appear above them in the t-direction.

The naive way of doing this would be to take the, at most, four polygons from one space-time prism, and, at most four, from the other and verify in they intersect. However, the nature of these polygons allow us to decide rather quickly which pair of polygons to dismiss and which pair needs further inspection. In the following, we assume to investigate the edge bounded by vertices r and s, where the arrival and departure times for the first space-time prism are t_r^-, t_r^+, t_s^- and t_s^+ respectively, these times for the second space-time prism will carry a prime, i.e., $t_r'^-, t_r'^+, t_s'^-$ and $t_s'^+$ respectively (in this section, we continue the terminology used in Section 5.2.1).

We apply the following two tests to all relevant line segments in the road network:

- if the time intervals are disjunct then exit and return False for this

segment. (This mean checking if either $t_r^- > t_r'^+ \ \wedge \ t_s^- > t_s'^+$ or $t_r^- < t_r'^+ \ \wedge \ t_s^- < t_s'^+$.);

- if the time intervals overlap then exit and return True for this segment. (This mean checking if either $t_r'^- \leq t_r^- \leq t_r'^+ \ \vee \ t_r'^- \leq t_r^+ \leq t_r'^+ \ \vee \ t_r^- \leq t_r'^- \leq t_r^+ \ \vee \ t_r^- \leq t_r'^+ \leq t_r^+$ or $t_s'^- \leq t_s^- \leq t_s'^+ \ \vee \ t_s'^- \leq t_s^+ \leq t_s'^+ \ \vee \ t_s^- \leq t_s'^- \leq t_s^+ \ \vee \ t_s^- \leq t_s'^+ \leq t_s^+$).

If the previous tests are inconclusive we will test the remaining cases with line-segment intersections using the formulas from Section 4.1.1. To do that, we will align the segment we are investigating on the x-axis and place one point in the origin to simplify computation: $x_0 = 0$ and $x_1 = d_{rs} = \sqrt{(x_s - x_r)^2 + (y_s - y_r)^2}$. Now assume $t_r^+ < t_r'^-$, if this is not the case then switch the role of the first and second space-time prism before proceeding. This also means $t_s^- > t_s'^+$. Let $\psi_\cap(x_0, t_0, x_1, t_1, x_0', t_0', x_1', t_1')$ be a formula that returns *true* if the segment bounded by the points (x_0, t_0) and (x_1, t_1) intersects the segment bounded by the points (x_0', t_0') and (x_1', t_1'), and that returns *false* if they do not intersect.

Now we can write the remaining steps of the algorithm.

1. If $r \in \mathsf{V}_\mathcal{P}$ and $s \in \mathsf{V}_{\mathcal{P}'}$ then let $t_2 = \frac{t_r^+ + t_r^-}{2}$, $x_2 = v_{rs}\frac{t_r^+ - t_r^-}{2}$, $t_2' = \frac{t_s'^+ + t_s'^-}{2}$ and $x_2' = x_1 - v_{rs}\frac{t_s'^+ - t_s'^-}{2}$. If $\psi_\cap(x_0, t_r^-, x_2, t_2, x_1, t_s'^-, x_2', t_2')$ or $\psi_\cap(x_0, t_r^+, x_2, t_2, x_1, t_s'^+, x_2', t_2')$ returns True, then return True and exit;

2. If $r \in \mathsf{V}_{\mathcal{P}'}$ and $s \in \mathsf{V}_\mathcal{P}$, then do the same as in step one but change the primes on the coordinates, i.e., give primes to the coordinates without primes and vice versa. Leave x_0 and x_1 unchanged.

3. If $t_s^+ \geq t_r^- + w_{rs}$ then let f be the segment (x_0, t_r^-, x_1, t_s^+) and if $t_r^+ \geq t_s^- + w_{rs}$ then let f be the segment (x_0, t_r^+, x_1, t_s^-); If $t_s'^+ \geq t_r'^- + w_{rs}$, then let f be the segment $(x_0, t_r'^-, x_1, t_s'^+)$ and if $t_r'^+ \geq t_s'^- + w_{rs}$ then let f be the segment $(x_0, t_r'^+, x_1, t_s'^-)$. If f or f' does not exist, then return False and exit; If $\psi_\cap(f, f')$ returns *true*, then return *true* and exit.

4. Return *false* and exit.

This concludes the algorithm to compute the answer to the alibi-query.

5.3.2 Complexity considerations

The formula we have given in Section 4.1.1, decides the emptyness of the intersection of two space-time prisms in constant time. To decide the alibi query for a lifeline necklace containing N space-time prisms and another consisting

of M space-time prisms, costs $\mathcal{O}(N + M)$ time (when first emptyness of the time intervals is tested).

For the space-time prism intersection on road networks, there is a fixed upper bound on tests that need to be done per edge that is in both space-time prisms. If the pre-computation is already done, then we first need to select the edges that appear in both projected ellipses of the constrained space-time prisms. This can be done, assuming once again that the vertices are ordered, in $\mathcal{O}(m + n)$ time where m is the number of vertices in the first space-time prism and n the number of vertices in other space-time prism.

Once this intersection is selected we iterate over every edge just once performing a number of intersection tests which has a fixed upper bound. The number of edges in the overlapping areas bounded by the ellipses are $\mathcal{O}(\mathsf{min}\{m, n\})$. This means we can conclude that deciding the emptiness of space-time prism intersection can be done in $\mathcal{O}(\mathsf{max}\{m, n\})$ time if the pre-computation is already done. It takes $\mathcal{O}\Big((\mathsf{max}\{m, n\})^2\Big)$ time if the pre-computation is not available.

To decide the alibi query for a space-time prism chain containing N space-time prisms and another consisting of M space-time prisms, we need to multiply this cost further by $\mathcal{O}(N + M)$.

5.4 Visualizing the intersection of two space-time prisms

The alibi query is a decision problem, which we solved in the previous section. For application purposes however, it is interesting to know, *if* two moving objects could have met, *where* they could have met, *when* and *for how long* they could have met and where they could have traveled together. All these questions, and more, can be answered by actually computing the intersection of the space-time prisms if this intersection is not empty.

This, in turn, is not too difficult since we already figured out how to compute single space-time prisms. Like in the decision problem, we iterate over every edge that has at least one polygon in both space-time prisms, and compute the intersection as an intersection between polygons. There is however, one small issue that needs to be considered. In the computation of the space-time prism polygons in Section 5.2, we used distances, which, were computed using square roots and this inevitably introduces rounding errors. The errors in turn cause polygons on the same edge to not always be aligned, i.e., being part of the same plane.

We solve this by projecting the polygons onto a two-dimensional space before computing intersections, similar to the approach we take in solving the

decision problem. Suppose we are projecting the polygons above the edge bounded by $r = (t_r, x_r, y_r)$ and $s = (t_s, x_s, y_s)$. Every corner-point $p = (t_p, x_p, y_p)$ of the polygons in mapped to a point $\left(t_p, \sqrt{(x_p - x_r)^2 + (y_p - y_r)^2}\right)$, thus effectively translating the polygons so that r becomes the origin and the edge is aligned with the x-axis.

Reconstructing the polygons is then fairly straightforward. Let (t_p, x_p) be a corner-point of the polygon that makes up the intersection, and $\overrightarrow{u_{rs}}$ be the unit vector derived from $\overrightarrow{rs}$. The point (t_p, x_p) is then mapped to the point $(t_p, x_r, y_r) + x_p.\overrightarrow{u_{rs}}$.

INTERSECTION VISUALIZATION: input= $(\mathsf{RN}, \mathcal{P}_1, \mathcal{P}_2)$;
output=drawing of $\mathcal{P}_1 \cap \mathcal{P}_2$.

Step 1. Select the edges that occur in both space-time prisms.

Step 2. Iterate over all the selected edges.

1. Project all the polygons onto the (x, y)-plane using the procedure described above.
2. Compute the intersections of every polygon on that edge belonging to $\mathcal{P}_1$ with every polygon belonging to $\mathcal{P}_2$.
3. Reconstruct the intersection as described above.

Note that for complexity considerations step 2 in the above algorithm can be executed in time $\mathcal{O}(1)$ since the number of polygons per space-time prism per edge is at most two and each polygon has at most four vertices.

Figures 5.11 and 5.12 are the output we created with our implementation. They show a yellow road network space-time prism and a blue one with different departure and arrival times and locations. The green coloured part depicts their intersection. We also produced a projection of the space-time prisms and their intersection on the road network in their respective colours. The vertical straight line on the left is the time axis. We also projected the intersection on that axis.

5.5 Example Queries

To show the usefulness of computing the intersection, we present some example queries. The above algorithms are cut out to provide answers to these queries.

- When Alfred leaves location p at time t_p and has to be at location q at time t_q, can Alfred visit location r and how long can Alfred stay there?

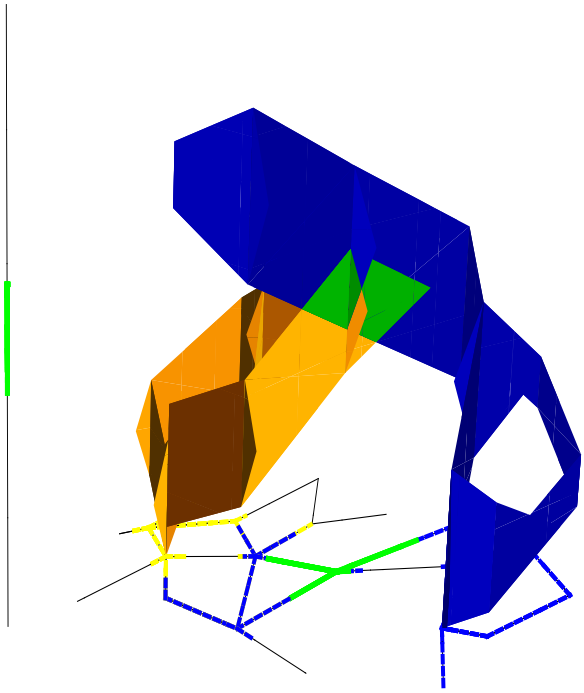


Figure 5.10: Example 1: Road Network space-time prisms with their intersection.

To answer this query we do not need space-time prisms just yet. We can compute the time of arrival and departure using network time at location r.

- When Alfred leaves location p at time t_p and has to be at location q at time t_q, how much time does Alfred have at most to spare?

- When Alfred leaves location p at time t_p and has to be at location q at time t_q, will Alfred still make it on time if the (speed limits of the) road network segments ... change, and how will that affect his time to spare?

These queries can be solved by computing and recomputing single space-time prisms from p to q and checking the difference between arrival and departure times at q.

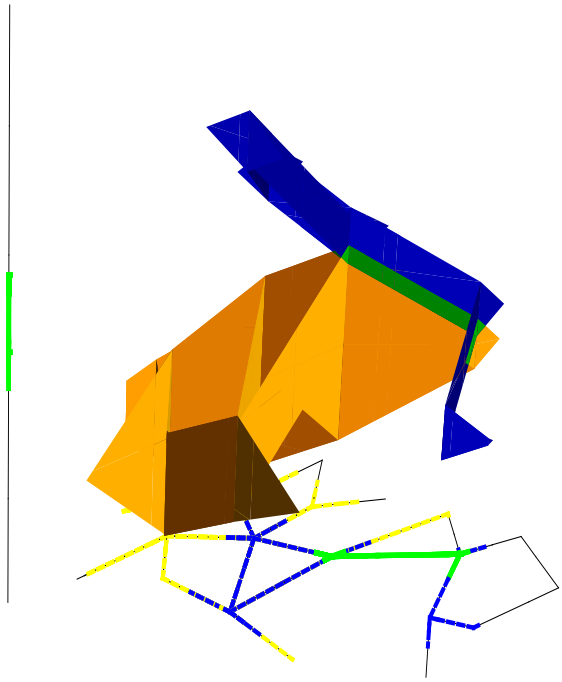


Figure 5.11: Example 1: Road Network space-time prisms with their intersection.

- When Alfred leaves location p at time t_p and has to be at location q at time t_q and James leaves location u at time t_u and has to be at location s at time t_s, when and where can they meet for at least 15 minutes?
- When Alfred leaves location p at time t_p and has to be at location q at time t_q and James leaves location u at time t_u and has to be at location s at time t_s, where can they both meet up at earliest (or latest) and go to location w, and how much time can they spend there?

To solve these queries we actually have to compute the space-time prisms and their intersection. Then use this intersection to answer the query.

- Alfred works at location p and has a 30 minutes lunch break and his walking speed is v, where can Alfred go eat (or is there a McDonalds

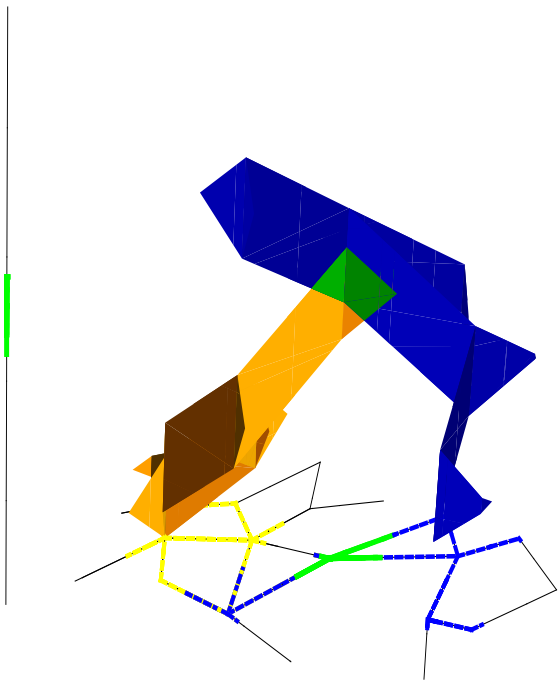


Figure 5.12: Example 2: Road Network space-time prisms with their intersection.

Alfred can reach) and spend at least 15 minutes there?

To answer this final query we compute the space-time prism with the same departure and arrival location. Then we check all food establishments in the projection of this prism and verify which establishment allows a 15 minutes stay.

The examples above show that these space-time prisms can answer real life questions that involve planning.

5.6 Conclusion and Future Work

In this chapter, we presented fast algorithms to compute and visualize space-time prisms on road networks where each edge in this network may have a

different speed limit. As proof of concept we implemented these algorithms in MATHEMATICA (available at [24]). We also presented an algorithm to decide the alibi query for two moving objects on a straight line and on a road network, respectively. The solution to this query turns out to be easier if the speed limit is tied to the road network and not to the moving objects. Furthermore, we concluded by presenting algorithms to visualize the intersection of two space-time prisms on a road-network.

The main simplification in our setting is the assumption that segments in the road network are bidirectional. We can modify our model to unidirectional segments by observing the following. In Theorem 5.3, we note that space-time prisms on road networks can be constructed by merely constructing the space-time prisms for each segment and each vertex separately. It is not hard to show that the space-time prisms in vertices, corresponding to Figure 5.4 and Cases 3 and 4 in Figure 5.5, reflect travel that is not constrained by direction on the segment. Whereas the space-time prisms on segments, corresponding to Case 5 in Figure 5.5, reflects all trajectories constrained by a mandatory direction from r to s on the segment.

In order to support unidirectional segments in our road network model, we have to introduce labels on segments indicating the allowed directions of movement. Our algorithm needs a slight modification in this setting. For bidirectional segments, we proceed as described before. For unidirectional segments, say from a vertex r to a vertex s where movement is only allowed from r to s, we omit constructing the polygons from Figure 5.4 and only construct the polygon as shown in Case 5 of Figure 5.5.

Note that once we allow unidirectional segments, a road network is not a metric space anymore. However, this is not a necessary condition to construct space-time prisms.

Obvious extensions to the work we presented, include but are not limited to, investigating efficient methods to update space-time prisms when adding or removing sample points. Another aspect to study is that of temporal changes in the road network, like draw bridges or railway crossings.

6

Uncertain space-time prisms

Space-time prisms capture all possible time-space locations of a moving object between sample points given speed limit constraints on its movement. These sample points are usually considered to be perfect measurements. In this chapter, we restrict ourselves to a road network and extend the notion of sample points to sample regions, which are bounded, sometimes disconnected, subsets of space-time wherein each point is a possible location, with its respective probability, where a moving object could have originated from or arrived in. This model allows us to model measurement errors, multiple possible simultaneous locations and even *flexibility* of a moving object.

We develop an algorithm that computes the *envelope* of all space-time prisms that have an anchor in these sample regions and we developed an algorithm that computes for any time-space point the probability with which a space-time prism, with anchors in these sample regions, contains that point. We implemented these algorithms in MATHEMATICA to visualise all these newly-introduced concepts.

6.1 From uncertainty on sample points to sample regions and uncertain space-time prisms

In the preliminaries, the sample points were considered exactly measured data and represented by space-time points. In real life however, a lot of sources introduce errors on sample points. Measurement errors, for one, are introduced when measuring locations using GPS, for example. When a human is asked to

keep track of its locations and time spent at those locations, errors get easily introduced, e.g., "I left work between 5 and 5:30 P.M.". Therefore, we extend the concept of certain sample points to sample regions.

6.1.1 Uncertain anchors

In this section, we expand our model to the use of sample regions to supersede the notion of sample points, but we will restrict ourselves to road networks. For simplicity, we will start by defining a sample region on a line segment and later expand this definition to road networks. Another simplification in our model is that time and space are considered independent from each other. This ensures that our sample regions are box-shaped in space-time and that our model behaves in a polygonal fashion. We coin this rectangle a *sample region*.

Moreover, in this sample region some subsets can be more likely than others. This can be described using probability functions. We refer to Figure 6.1 for a conceptual representation of this model.

Definition 6.1. A *sample region on a segment* is a bounded subset of space-time of all possible locations of a sample point. Let $p = (x_p, y_p), q = (x_q, y_q) \in \mathbf{R}^2$ be the spatial points bounding the segment and $t_{pq}^-, t_{pq}^+ \in \mathbf{R}$, where $t_{pq}^- \leq t_{pq}^+$, the temporal boundaries of the region, meaning the region is bounded by the rectangle $\langle (t_{pq}^-, x_p, y_p), (t_{pq}^+, x_p, y_p), (t_{pq}^+, x_q, y_q), (t_{pq}^-, x_q, y_q) \rangle$. We describe this region by the 6-tuple $S_{pq} = (p, q, t_{pq}^-, t_{pq}^+, \mu_{pq}, \chi_{pq})$, where $\chi_{pq} : \mathbf{R} \to \mathbf{R}^+$ and $\mu_{pq} : [t_{pq}^-, t_{pq}^+] \to \mathbf{R}^+$ are independent probability functions and $\chi_{pq} : \lambda \mapsto \chi_{pq}(\lambda)$ where χ_{pq} is a probability function on the line $(x, y) = (1-\lambda)p + \lambda q$. □

We note that in the definition above we do not demand that χ_{pq} is restricted to $[0, 1]$. The reason for this is that on a road network RN we will stitch a finite number of these previously defined sample regions together and we want these probability functions to integrate to one on all these regions combined.

Definition 6.2. A *sample region S on a road network* RN is a bounded subset of space-time of all possible locations of a sample point on RN. In particular, it is a finite set of *sample regions on segments* as defined in Definition 6.1, $S = \cup_{i=1}^{n} S_i$ where each S_i is a sample region on a segment. Moreover, all the S_i are disjoint and integrating all the spatial probability functions over all the spatial component of these sample regions adds up to one.

An *uncertain trajectory sample* is then a finite list of sample regions and constitutes a new definition of trajectory samples.

□

This is illustrated in Figure 6.1.

We remark the following.

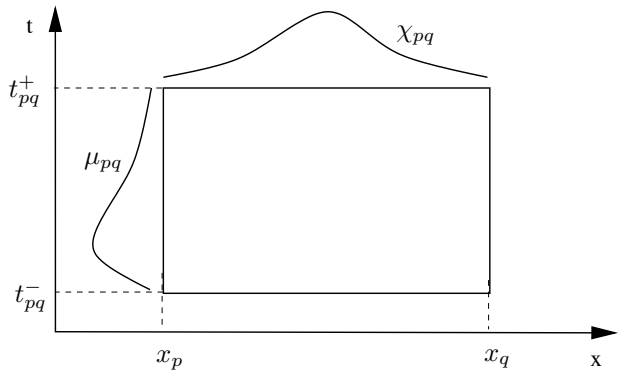


Figure 6.1: An example of a sample region with probability functions χ_{pq} and μ_{pq}.

- We are not demanding that the regions stitch nicely together in time. This allows us to model locations that are accessible at discrete intervals in time. For example, shopping malls that are not open 24 hours a day.
- Secondly, we do not demand that they are continuous in space either. This, in turn, allows us to model several probable departure and arrival locations, and, as we will soon elaborate on, calculate a relative likelihood for each region.

6.1.2 Uncertain space-time prisms

The next step is to adapt the space-time prism model to these sample regions. This can be done in a straightforward manner as described in the next definition.

Definition 6.3. Let RN be a road network and S_b and S_e sample regions on RN. An *uncertain space-time prism* between the sample regions S_b and S_e is the union of all space-time prisms with starting point in S_b and ending point in S_e. An additional constraint is that there needs to exist a space-time prism from every point in S_b that has an endpoint in S_e and vice versa. □

The last condition ensures that our model makes sense. We want to model possible locations but if there exists a point in a sample region that is not an anchor for a space-time prism with an anchor in the next region, then the moving object could not possibly have departed from that point. In that case a distribution that is nonzero in that point does not make sense.

When every point in a sample region is equally probable then the distribution function for time and space is a uniform distribution. To model those

points that are more probable around the center of a sample region than at the edges, a normal distribution can be used.

6.2 Computing the envelope of the uncertain prism

In this section we introduce an algorithm that computes the uncertain space-time prism, which envelopes the union of all space-time prisms that connect a point from the starting sample regions to a point in the ending sample regions. This algorithm is a slight adaptation of the algorithm we present in Chapter 5. where also the proofs of the correctness of this algorithm can be found. We consider each edge separately and it suffices that we compute, for both vertices of that edge, the earliest arrival times and latest departure times.

The following observations will simplify the computations significantly. For each edge, where we are constructing the uncertain prism, we cycle through all the edges of the starting sample region. For each such edge we note that the fastest path has to pass over one of the nodes of that edge. Moreover, a path from the interior of such an edge is longer than a path that departs from at least one of the nodes. So the earliest time you can reach a node is the minimum of the road network distances to that node from all nodes of the departure sample region at the earliest time of that region. Likewise, the latest departure times at each node will equal the maximum of all road network distances to all nodes of the arrival sample region at the latest time of that region.

The major difference with the algorithm in Chapter 5 is the pre-computation part of the algorithm. In the first step, we use an adapted breadth-first search to pre-select the vertices that can be part of the space-time prism and ignore the rest. In the second step we compute the earliest arrival and latest departure times for each vertex. This computation is not sufficient for the edges that support sample regions, since these computations that do not take the points in the interior into account.

PRECOMP: input= $(\mathsf{RN}, \mathsf{V}, \mathsf{E}, S_b, S_e)$;
output= $(\mathsf{RN}', \{(t_u^-, t_u^+) \mid u \in \mathsf{RN}'\}, \mathsf{V}_\mathcal{P}, \mathsf{E}_\mathcal{P})$

Step 1. In this step we add vertices to the network and select those nodes and vertices that can actually support the prism.

- For all $S_{b,i}, S_{e,j}$, say such a region is spatially bounded by $p, q \in \mathbf{R}^2$. If p is a vertex, then do nothing, else let $r, s \in \mathbf{R}^2$ be the vertices that bound the edge that contains p. Remove that edge from the network, add the vertex p to the network and add the edges bounded by r and p and the

edge bounded by p and s to the network. Repeat the same procedure for q.

- The second part of this step consists of an adapted breadth-first search algorithm where we keep looking for and storing vertices and edges until their road network distance to any $S_{b,i}$ or $S_{e,j}$ is larger than the maximal difference in time between any pair $(S_{b,i}, S_{e,j})$. Let $t_{\max} = \max_{i,j}\{t^+_{e,j} - t^-_{b,i} \mid S_{b,i} = (p_{b,i}, q_{b,i}, t^-_{b,i}, t^+_{b,i}, \mu_{b,i}, \chi_{b,i})$ and $S_{e,j} = (p_{e,j}, q_{e,j}, t^-_{e,j}, t^+_{e,j}, \mu_{e,j}, \chi_{e,j})\}$. The following steps need to be repeated for all $\{r \mid \exists i : S_{b,i} = (p, q, t^-, t^+, \mu, \chi)$ or $\exists j : S_{e,j} = (p, q, t^-, t^+, \mu, \chi)$ and $r = p$ or $r = q\}$. Initialise a queue with the node r and distance 0. Add r to the road network RN'. Repeat the following steps until the queue is empty.

  1. Remove the top element from the queue, which is a vertex s and a distance t_s.
  2. For all the edges that s to a vertex p, do the following:
     - If p has not yet been handled, then add the vertex p and the edge connecting p and s to RN' and mark this edge and vertex as handled.
     - If p has not been handled yet and $t_s + \mathsf{d}_{\mathsf{RN}}(s, p) \leqslant t_{\max}$ then add the vertex p and the distance $t_s + \mathsf{d}_{\mathsf{RN}}(s, p)$ to the queue.
     - If p has been handled and $t_s + \mathsf{d}_{\mathsf{RN}}(s, p) \leqslant t_{\max}$ and $t_s + \mathsf{d}_{\mathsf{RN}}(s, p) \leqslant t_p$ where t_p is a previously recorded distance for p, then add the vertex p and the distance $t_s + \mathsf{d}_{\mathsf{RN}}(s, p)$ to the queue.

Step 2. In this step we compute the earliest arrival time and latest departure time in each vertex, with respect to all the sample regions where we can leave from and all regions where we can arrive in.

Let $V_b = \{(r, t^-) \mid \exists i : S_{b,i} = (p, q, t^-, t^+, \mu, \chi)$ and $r = p$ or $r = q\}$ and $V_e = \{(r, t^+) \mid \exists j : S_{e,j} = (p, q, t^-, t^+, \mu, \chi)$ and $r = p$ or $r = q\}$.

Now we cycle through all the pairs (r, t^-) contained in V_b and apply a single-source shortest path algorithm (e.g., Dijkstra's algorithm) for each r on the graph RN'. For each vertex u in RN' we store its smallest road network time from r, i.e., the arrival time $t^-_u = t^- + \mathsf{d}_{\mathsf{RN}}(r, u)$. For the first node r we initialise the nodes u with that arrival time, for all other nodes r we set the arrival time t^-_u to $\min\{t^-_u, t^- + \mathsf{d}_{\mathsf{RN}}(r, u)\}$.

Again, we cycle through all the pairs (r, t^+) contained in V_e and apply a single-source shortest path algorithm (e.g., Dijkstra's algorithm) for each r on the graph RN'. For each vertex u in RN' we store its largest road network time from r, i.e., the arrival time $t^+_u = t^+ - \mathsf{d}_{\mathsf{RN}}(r, u)$. For the first node r we

initialise the nodes u with that arrival time, for all other nodes r we set the arrival time t_u^+ to $\max\{t_u^+, t^+ - \mathsf{d}_{\mathsf{RN}}(r,u)\}$.

When we make our last pass for the last element of V_e, we store each vertex u for which $t_u^- \leq t_u^+$ in the set $\mathsf{V}_{\mathcal{P}}$. In the set $\mathsf{E}_{\mathcal{P}}$ we store all edges that connect to at least one vertex in $\mathsf{V}_{\mathcal{P}}$.

The next step is to construct the polygons that constitute the space-time prism. This step is identical to the algorithm 3D-SPACE-TIME PRISM described in Chapter 5. Since we already provided a correctness proof in that same chapter we will omit to repeat this here. The only difference is that we need to correct the polygons for the sample regions since our computations neglected the internal points of the sample region. An example of this is illustrated in Figure 6.2.

Figure 6.2: The omitted part of a sample region.

We can correct this by adding an extra, easy to compute, triangular polygon for each sample region. For each starting region $S_{b,i} = (p, q, t^-, t^+, \mu, \chi)$ we add the polygon $\langle (p, t^-), (q, t^-), (\frac{p+q}{2}, \frac{\mathsf{d}_{\mathsf{RN}}(p,q)}{2}) \rangle$ as illustrated in Figure 6.2 on the right. Likewise, for each ending region $S_{e,j} = (p, q, t^-, t^+, \mu, \chi)$ we add the polygon $\langle (p, t^+), (q, t^+), (\frac{p+q}{2}, -\frac{\mathsf{d}_{\mathsf{RN}}(p,q)}{2}) \rangle$.

The result of this pre-computation algorithm, together with the 3D-SPACE-TIME PRISM-algorithm is shown in Figure 6.3. We note that unlike in Figure 5.2, where a prism starts and ends in a single point, this prism has a lot of possible locations to start from and to arrive at.

6.3 Measuring spatio-temporal uncertainty and flexibility with respect to sample regions

In this section, we go a step further and introduce the main contribution of this paper. When we introduced sample regions, we used distributions functions

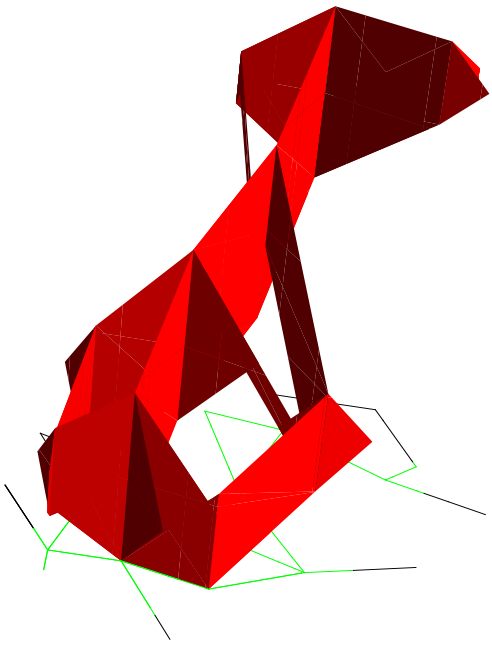


Figure 6.3: An envelope of space-time prisms on sample regions.

to model the likelihood of every point in the sample region, but we have yet to exploit those attributes of sample regions.

Let r be any time-space point inside the envelope of the uncertain space-time prism. We know that r is covered by at least one space-time prism with a starting point in the starting region and an ending point in the ending region. Suppose, and this is the case for most points in the envelope, that there is an entire subset of the sample regions consisting of points which are anchors for *space-time prisms that contain* r. We can then *measure*, i.e., integrate the distribution functions over these subsets by means of the distribution functions we introduced in Definition 6.1, and we can choose to

(1) restrict ourselves to the starting regions,

(2) restrict ourselves to the ending regions,

(3) multiply the two numbers above.

In all three cases we get a number between zero and one. This is per construc-

tion of the sample regions.

In Case (1) we obtain the likelihood that the anchor of a space-time prism that contains r is part of the starting regions. In Case (2) we obtain the likelihood that the anchor of a space-time prism that contains r is part of the ending regions. In Case (3) we obtain the simultaneous likelihood that the anchor of a space-time prism that contains r is part of the starting and ending regions. The fraction of the sample regions that have a starting and ending point of a space-time prism that contains that particular space-time point r.

Definition 6.4. Let r be a time-space point on a road network RN and S_b, S_e sample regions on RN.

- *The emanating fraction of r with respect to S_b* equals the measure, with respect to the distribution functions of S_b, of the subsets of S_b that contain anchors p, with a *smaller* time coordinate than r of space-time prisms on RN with anchors p and r.
- *The absorbing fraction of r with respect to S_e* equals the measure, with respect to the distribution functions of S_e, of the subsets of S_e that contain anchors q, with a *larger* time coordinate than r of space-time prisms on RN with anchors r and q.
- *The combined fraction of r with respect to travel from S_b to S_e* equals the product of (the emanating fraction of r with respect to S_b) with (the absorbing fraction of r with respect to S_e).

□

In the following section we show the surprisingly simple, i.e., polygonal, shape of these subsets. Moreover we provide an algorithm to compute these subsets and associated fractions as defined in Definition 6.4.

6.3.1 Algorithms

The algorithm we present here is based on the following observations, which do not require proof.

Due to the additive nature of integrals, i.e., an integral over a surface equals the sum of integrals over disjoint subsets that cover the surface, we can treat each of the rectangular regions in space-time separately and add them together once the computation is done. A sample region can thus be seen as the sum of sample regions on straight edges of the road network.

Let $r = (t_r, x_r, y_r)$ be the time-space point for which we wish to compute the fraction of space-time prisms that contain that point. For each of those separate rectangular sample regions, we can distinguish between two cases.

Either there exists a point s in the spatial part of the sample region for which there exists more than one fastest path to r, or there does not. If this point exists we simply divide the rectangular region into two new regions along the temporal line through s. This operation ensures we are again in the latter case, where there exists no spatial point, in the interior of the region, for which there exists more than one fastest path to r. See Figure 6.4.

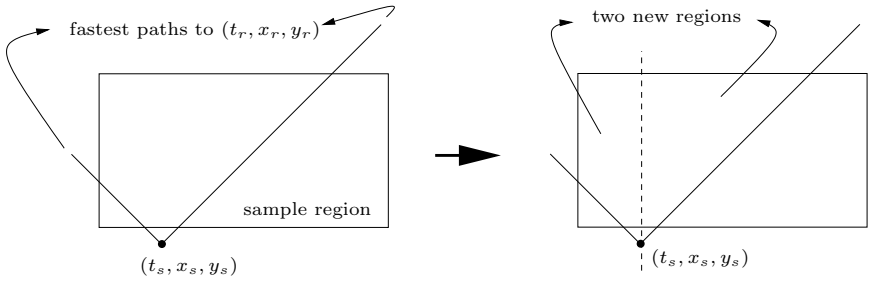


Figure 6.4: A divided sample region.

The second observation, which we will prove in Theorem 6.5, states that it suffices to find all space-time paths that originate in the starting sample region and end in r, and all space-time paths that originate in r and arrive in the ending sample region.

The proof of the following theorem is trivial, the equivalence holds by the very definition of a space-time prism.

Theorem 6.5. *Let* RN *be a road network, p and q points on* RN *and r a time-space point. There exists a space-time prism $\mathcal{P}^{\mathsf{RN}}(t_p, p, t_q, q)$ that contains r* if and only if *there exists a trajectory from (t_p, p) to (t_q, q) on* RN *through r.* □

Though the proof is trivial, it holds the key to our algorithm, which is based on the following two observations. Assume that the interior of the sample region S_b does not contain a point with more than one path with the same road network time to r. We will show in the algorithm below how to split the sample regions to smaller regions that satisfy that condition.

1. First, a fastest path that leads to r and contains the edge contained by S_b either intersects S_b, goes over S_b, i.e., all its time coordinates are larger than those of S_b, or under S_b, i.e., all its time coordinates are smaller than those of S_b. If it goes over S_b then all points of S_b clearly have a path to r, since any moving object in a point in S_b can just wait until its time coordinate equals that of the fastest path and leave to r following that path. Likewise, if the path goes under S_b then no moving object departing from S_b is able to reach r in time. If the path intersects S_b,

then all space-time points of S_b with a time coordinate smaller than the point on the path with the same spatial coordinates have a path that reaches r in time. This is depicted in the shaded region in Figure 6.5(a).

2. Secondly, a fastest path that emanates from r and contains the edge contained by S_e either intersects S_e, goes over S_e or under S_e. If it goes under S_e, i.e., all its time coordinates are smaller than those of S_e, then all points of S_e can be reached from r, since any moving object that departs from r can reach a point with spatial coordinates in S_e and wait until its time coordinate equals that of a point in S_e. Likewise, if the path goes over S_e, i.e., all its time coordinates are greater than those of S_e, then no moving object departing from r is able to reach S_e in time. If the path intersects S_e, then all space-time points of S_e with a time coordinate greater than the point on the fastest path with the same spatial coordinates have a path from r that can be reached in time. This is depicted in the shaded region in Figure 6.5(b).

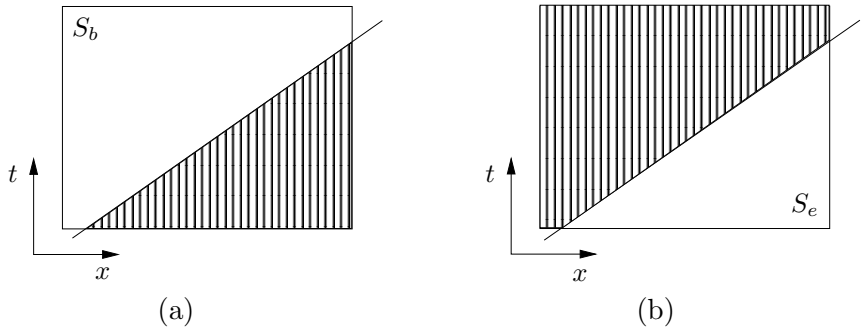


Figure 6.5: Subsets of sample regions that are possible to connect to a fixed space-time point.

These observations are necessary and sufficient to compose the algorithm. Once these areas have been determined we can integrate the distribution functions over them and compute a probability for a specific space-time point. The nature of these areas allow easy computation of these integrals.

Let $r = (t_r, x_r, y_r)$ be the space-time point for which we want to compute the fraction of space-time prisms covering it. Let S_b be the starting sample region and S_e be the ending sample region. Let $S_{b,i}$ be the restriction of S_b to a single edge and i be a natural number to count the number sample regions on a segment that S_b contains, the definition of $S_{e,i}$ is analogous. The first

step in the algorithm is to compute appropriately sized sample regions. These are regions where each point in the spatial interior has a unique fastest path to the given space-time point r.

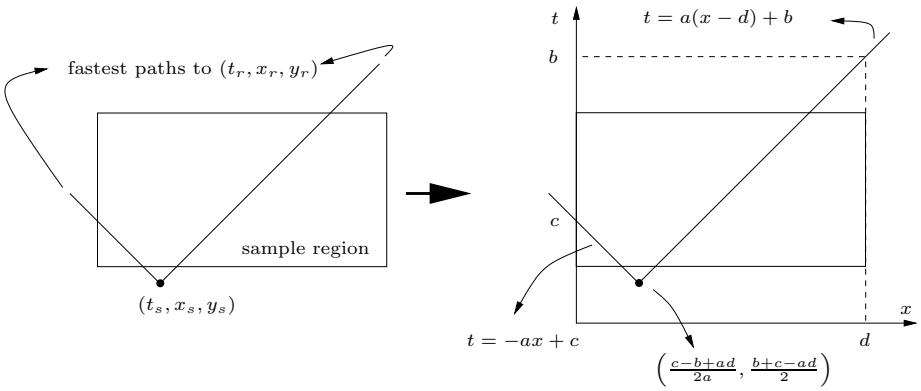


Figure 6.6: How to split a sample region.

The following algorithm takes a time-space point $r = (t_r, x_r, y_r)$ as input and outputs three *numbers*: the emanating fraction $\mathcal{S}_{b,r}$ of r with respect to S_b, the absorbing fraction $\mathcal{S}_{e,r}$ of r with respect to S_e and the fraction $\mathcal{S}_r$ of r with respect to travel from S_b to S_e.

PointProbability: input= $(\mathsf{RN}, \mathsf{V}, \mathsf{E}, t_r, x_r, y_r, S_b, S_e)$;
output= $(\mathcal{S}_{b,r}, \mathcal{S}_{e,r}, \mathcal{S}_r)$

Initialisation. Set $\mathcal{S}_{b,r}$ and $\mathcal{S}_{e,r}$ equal to 0.

Step 1. Let $S_{b,i} = (p, q, t^-_{pq}, t^+_{pq}, \mu_{pq}, \chi_{pq})$ where $p = (x_p, y_p)$ and $q = (x_q, y_q)$. The following needs to be repeated for each $S_{b,i}$. Let $d_p = t_r - \mathsf{d}_{\mathsf{RN}}((x_r, y_r), (x_p, y_p))$, $d_q = t_r - \mathsf{d}_{\mathsf{RN}}((x_r, y_r), (x_q, y_q))$, $d_{pq} = \mathsf{d}_{\mathsf{RN}}((x_p, y_p), (x_q, y_q))$ and v_{pq} be the speed limit on the segment that supports $S_{b,i}$.

- **Case 1:** $d_p \pm d_{pq} = d_q$**.**
  - If $d_p - d_{pq} = d_q$, then interchange the roles of p and q, now we have $d_p + d_{pq} = d_q$.
  - If $d_q \leqslant t^-_{pq}$, then do nothing and move on to the next $S_{b,i}$. See Figure 6.7 on the left.
  - If $d_p \geqslant t^+_{pq}$, then let replace $\mathcal{S}_{b,r}$ by $\mathcal{S}_{b,r} + \int_0^1 \chi_{pq}$ and move on to the next $S_{b,i}$. See Figure 6.7 on the right.
  - Else If $d_p \leqslant t^-_{pq}$, then let $\mathcal{A}$ be the area bounded by the polygon $\langle (d_p, x_p, y_p), (d_p, x_q, y_q), (d_q, x_q, y_q) \rangle$ **else** let $\mathcal{A}$ be the area bounded

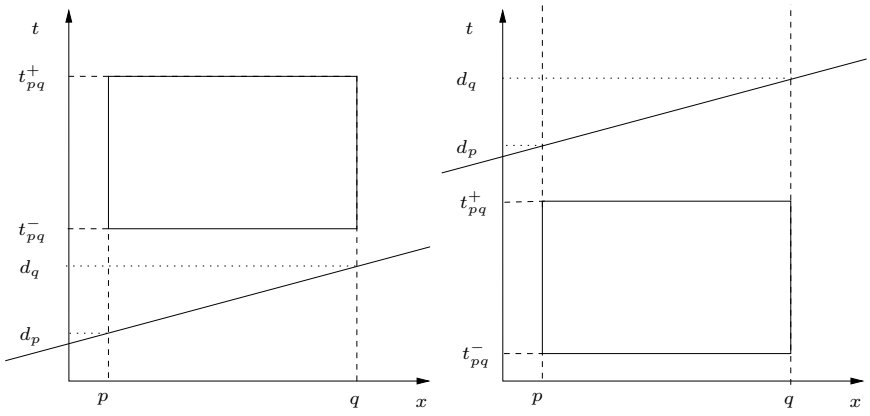


Figure 6.7: Case illustration.

by the polygon $\langle (d_p, x_p, y_p), (t^-_{pq}, x_p, y_p), (t^-_{pq}, x_q, y_q), (d_q, x_q, y_q)\rangle$. Replace $\mathcal{S}_{b,r}$ by $\mathcal{S}_{b,r} + \int\int_{\mathcal{A}} \mu_p \chi_p$. See Figure 6.8 on the left for the first polygon, and on the right for the second polygon.

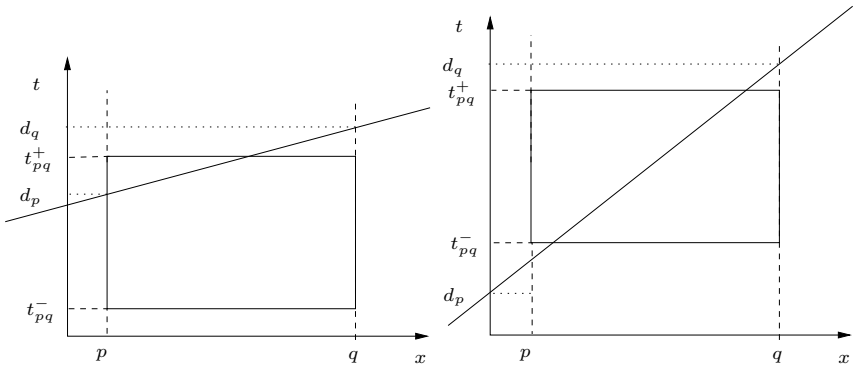


Figure 6.8: Case illustration.

We note that although $\mathcal{A}$ is likely to exceed the boundary of $S_{b,i}$, the integral is still well defined because μ_p is zero outside $S_{b,i}$.

- **Case 2:** $d_p \pm d_{pq} \neq d_q$

  If this is the case we have to compute the point on the segment where the two fastest paths from this segment to r intersect and distinguish three separate sub-cases. The first is the easiest, when $d_p, d_q \leqslant t^-_{pq}$ then this inequality will also hold for the time-coordinate where the two fastest

paths from r intersect, in that case none of the points from $S_{b,i}$ will be able to reach r in time.

– If $d_p, d_q \leqslant t^-_{pq}$, then do nothing and proceed to the next sample region. See Figure 6.9 on the left.

If all time coordinates satisfy $\frac{d_p+d_q-d_{pq}}{2}, d_p, d_q \geqslant t^+_{pq}$ then all points of $S_{b,i}$ will be able to reach r in time. In that case it is pointless to compute intersections and divide the sample region. Hence,

– If $\frac{d_p+d_q-d_{pq}}{2}, d_p, d_q \geqslant t^+_{pq}$, then replace $\mathcal{S}_r$ by $\mathcal{S}_r + \int_0^1 \chi_{pq}$ and proceed to the next sample region. See Figure 6.9 on the right.

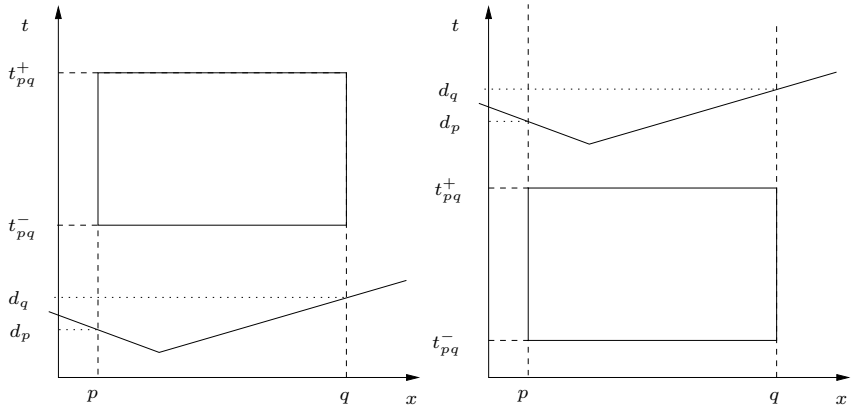


Figure 6.9: Case illustration.

In the remaining case one of the fastest paths from r intersects $S_{b,i}$ and we need to split $S_{b,i}$ into two regions such that all points in those regions have a unique fastest path to r. We apply the same strategy as in Chapter 5. As shown in Figure 6.6, we merely need to reduce our computations to the two-dimensional case, compute the x-coordinate of the intersection and multiply that by an appropriate unit vector. In this case $S_{b,i}$ will be split at the spatial point

$$s = (x_s, y_s) = (x_p, y_p) + \left(\frac{d_p - d_q + d_{pq}}{2/v_{pq}}\right) \cdot \frac{(x_q - x_p, y_q - y_p)}{\sqrt{(x_q - x_p)^2 + (y_q - y_p)^2}}.$$

– Else Replace $S_{b,i}$ by the two new regions $S'_{b,i}$ and $S''_{b,i}$ where
- $S'_{b,i} = (p, s, t^-_{pq}, t^+_{pq}, \mu_{pq}, \chi_{pq} \circ f)$ where

$$f(\lambda) = \lambda \cdot \sqrt{\frac{(x_s - x_p)^2 + (y_s - y_p)^2}{(x_q - x_p)^2 + (y_q - y_p)^2}};$$

- $S''_{b,i} = (s, q, t^-_{pq}, t^+_{pq}, \mu_{pq}, \chi_{pq} \circ g)$ where

$$g(\lambda) = \lambda \cdot \sqrt{\frac{(x_q - x_s)^2 + (y_q - y_s)^2}{(x_q - x_p)^2 + (y_q - y_p)^2}} + \sqrt{\frac{(x_s - x_p)^2 + (y_s - y_p)^2}{(x_q - x_p)^2 + (y_q - y_p)^2}};$$

and store d_p for the vertex (x_p, y_p), d_q for the vertex (x_q, y_q) and $\frac{d_p + d_q - d_{pq}}{2}$ in the vertex (x_s, y_s) to avoid re-computation in Case 1. Now proceed as in Case 1 for each of these two new regions.

Step 2. *The procedure $S_{e,i}$ is very much like the one outlined in Step 1, except, as indicated in Figure 6.5, we need to construct the polygons on the other side of the fastest path.*

Let $S_{e,i} = (p, q, t^-_{pq}, t^+_{pq}, \mu_{pq}, \chi_{pq})$ where $p = (x_p, y_p)$ and $q = (x_q, y_q)$. The following needs to be repeated for each $S_{e,i}$. Let $d_p = t_r + \mathsf{d_{RN}}((x_r, y_r), (x_p, y_p))$, $d_q = t_r + \mathsf{d_{RN}}((x_r, y_r), (x_q, y_q))$, $d_{pq} = \mathsf{d_{RN}}((x_p, y_p), (x_q, y_q))$ and v_{pq} be the speed limit on the segment that supports $S_{e,i}$.

- **Case 1:** $d_p \pm d_{pq} = d_q$
  - If $d_p - d_{pq} = d_q$, then interchange the roles of p and q, now we have $d_p + d_{pq} = d_q$.
  - If $d_p \geqslant t^+_{pq}$, then do nothing and move on to the next $S_{e,i}$.
  - If $d_q \leqslant t^-_{pq}$, then let replace $\mathcal{S}_{e,r}$ by $\mathcal{S}_{e,r} + \int_0^1 \chi_{pq}$ and move on to the next $S_{e,i}$.
  - Else If $d_q \geqslant t^+_{pq}$, then let $\mathcal{A}$ be the area bounded by the polygon $\langle (d_p, x_p, y_p), (d_q, x_p, y_p), (d_q, x_q, y_q) \rangle$ **else** let $\mathcal{A}$ be the area bounded by the polygon $\langle (d_p, x_p, y_p), (t^+_{pq}, x_p, y_p), (t^+_{pq}, x_q, y_q), (d_q, x_q, y_q) \rangle$. Replace $\mathcal{S}_{e,r}$ by $\mathcal{S}_{e,r} + \int\int_{\mathcal{A}} \mu_p \chi_p$.

  We note that although $\mathcal{A}$ is likely to exceed the boundary of $S_{e,i}$, the integral is still well defined because either μ_p is zero outside $S_{e,i}$.

- **Case 2:** $d_p \pm d_{pq} \neq d_q$

  If this is the case we have to compute the point on the segment where the two fastest paths from this segment to r intersect and distinguish three separate sub-cases. The first is the easiest, when $d_p, d_q \leqslant t^-_{pq}$ then this inequality will also hold for the time-coordinate where the 2 fastest paths from r intersect, in that case none of the points from $S_{b,i}$ will be able to reach r in time.
  - If $d_p, d_q \geqslant t^+_{pq}$, then do nothing and proceed to the next sample region.

If all time coordinates satisfy $\frac{d_p+d_q+d_{pq}}{2}, d_p, d_q \leqslant t^-_{pq}$ *then any moving point starting from* r *will be able to reach all points in* $S_{e,i}$ *in time. In that case it is pointless to compute intersections and divide the sample region. Hence,*

– If $\frac{d_p+d_q+d_{pq}}{2}, d_p, d_q \leqslant t^+_{pq}$, then replace $\mathcal{S}_{e,r}$ by $\mathcal{S}_{e,r} + \int_0^1 \chi_{pq}$ and proceed to the next sample region.

In the remaining case one of the fastest paths from r intersects $S_{e,i}$ and we need to split $S_{e,i}$ into two regions such that all points in those regions have a unique fastest path to r. We apply the same strategy as in Chapter 5. As illustrated in Figure 6.6, we merely need to reduce our computations to the two-dimensional case, compute the x-coordinate of the intersection and multiply that by an appropriate unit vector. In this case $S_{e,i}$ will be split at the spatial point

$$s = (x_s, y_s) = (x_p, y_p) + \left(\frac{d_q - d_p + d_{pq}}{2/v_{pq}}\right) \cdot \frac{(x_q - x_p, y_q - y_p)}{\sqrt{(x_q - x_p)^2 + (y_q - y_p)^2}}.$$

– Else Replace $S_{e,i}$ by the two new regions $S'_{e,i}$ and $S''_{e,i}$ where
- $S'_{e,i} = (p, s, t^-_{pq}, t^+_{pq}, \mu_{pq}, \chi_{pq} \circ f)$ where

$$f(\lambda) = \lambda \cdot \sqrt{\frac{(x_s - x_p)^2 + (y_s - y_p)^2}{(x_q - x_p)^2 + (y_q - y_p)^2}};$$

- $S''_{e,i} = (s, q, t^-_{pq}, t^+_{pq}, \mu_{pq}, \chi_{pq} \circ g)$ where

$$g(\lambda) = \lambda \cdot \sqrt{\frac{(x_q - x_s)^2 + (y_q - y_s)^2}{(x_q - x_p)^2 + (y_q - y_p)^2}} + \sqrt{\frac{(x_s - x_p)^2 + (y_s - y_p)^2}{(x_q - x_p)^2 + (y_q - y_p)^2}};$$

and store d_p for the vertex (x_p, y_p), d_q for the vertex (x_q, y_q) and $\frac{d_p+d_q-d_{pq}}{2}$ in the vertex (x_s, y_s) to avoid re-computation in Case 1. Now proceed as in Case 1 for each of these two new regions.

Output $\mathcal{S}_{b,r}$, $\mathcal{S}_{e,r}$ and $\mathcal{S}_r = \mathcal{S}_{b,r} \cdot \mathcal{S}_{e,r}$.

Now that we have an algorithm for an individual point, we can construct one to visualise this for all points of an uncertain space-time prism envelope. However, we will not do this for all points, we divide the envelope in smaller regions by overlaying it with a regular grid, pick a representative point for each region, compute its probability and assign a suitable colour to that entire region.

The 3D-SPACE-TIME PRISM algorithm outputs a set of polygons. Again, we cycle over all the edges of the road network that contain such a polygon. For each such edge we intersect the polygons with a two-dimensional grid on that edge, the size of the grid depends on a pre-chosen resolution. The intersection gives again a set of polygons, where no polygon is larger than the cells of the grid. For each of those we compute the center of mass which we will feed to our POINTPROBABILITY-algorithm. This in turn yields a number, between zero and one, which we associate with that center of mass and its polygon and use it to assign an appropriate colour to the polygon.

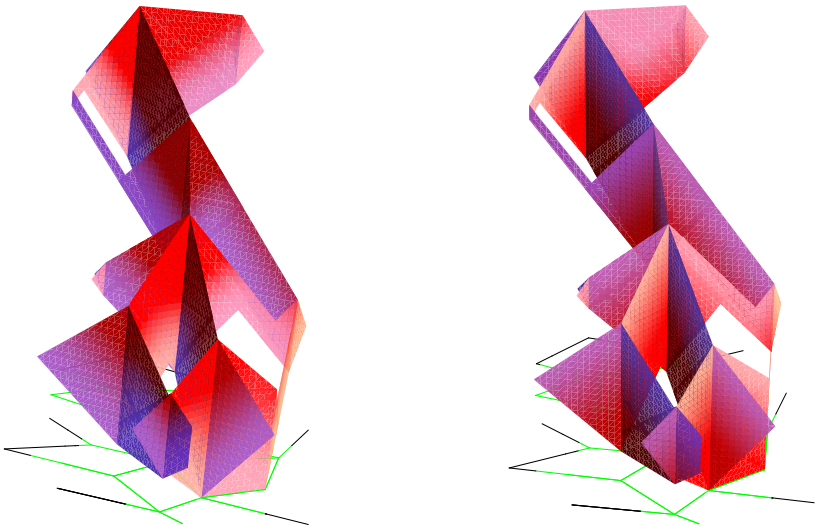


Figure 6.10: A space-time prism with the emanating fraction (left) and absorbing fraction (right) coloured in shades of red. The closer the colour comes to red, the higher the value of the fractions.

We implemented these algorithms in MATHEMATICA as a proof-of-concept. In Figure 6.10 on the left, we restricted our algorithm to compute the emanating fraction of each time-space point. In Figure 6.10 on the right we restricted our algorithm to compute the absorbing fraction of each time-space point. In Figure 6.11 our algorithm computed the emanating fraction of each time-space point with respect to travel from the originating regions to the destination region. In all these examples we assumed a uniform distribution on the sample regions.

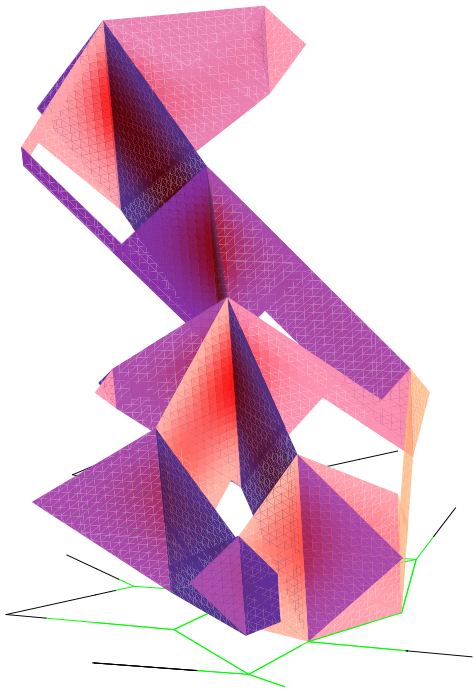


Figure 6.11: A space-time prism with the fraction of each space-time point coloured in shades of red.

6.4 Applications

In the previous sections, we introduced a multitude of new concepts and quantities. Each of those separately and combined can be used in a number of applications which we outline here. Such applications include measuring *flexibility* and *error analysis*.

6.4.1 Measuring flexibility

At the end of the previous section, we generated a fully coloured prism where all time-space points indicated their fraction simultaneously.

Assume that in a point r the emanating fraction of r with respect to S_b, as defined in Definition 6.4, is close to one. This means that we can reach r from most of S_b. More importantly, this means there exists a path from S_b that reaches the spatial component of r and allows us to spend a time at this location that equals almost all of the temporal height of S_b. We illustrate this

in Figure 6.12. The shaded part of S_b covers almost all of S_b, which means the emanating fraction of r with respect to S_b is close to one. A fastest path from any point on the bottom of S_b to the spatial location of r would be parallel in space-time to the fastest path drawn in Figure 6.12. Moreover, suppose the temporal height of S_b equals $\triangle t$, then any such path to the right of s from the bottom of S_b arrives at least $\triangle t$ earlier at the spatial location of r, we can thus spend at least $\triangle t$-time at the spatial location of r. If we leave from the left of s we have a little less than $\triangle t$-time to spend. So, the higher the emanating fraction, the more flexibility we have to choose when and where to leave from S_b and vice versa.

Likewise, assume that in a point r the absorbing fraction of r with respect to S_e, as defined in Definition 6.4, is close to one. This means that we can reach most of S_e from r. More importantly, we can spend an amount of time, that equals almost all of the temporal height of S_e, at the spatial component r before we have to leave again and still be able to reach S_e in time.

This gives us a measure of flexibility we have with respect to the starting and ending regions at each location at each moment in time. A more practical way is to apply a kind of spatial projection on the road network. If we project, for each location, the maximum of these fractions in time onto the road network, we immediately obtain the flexibility we have to reach those locations with respect to our schedule or probable locations to leave from or arrive at.

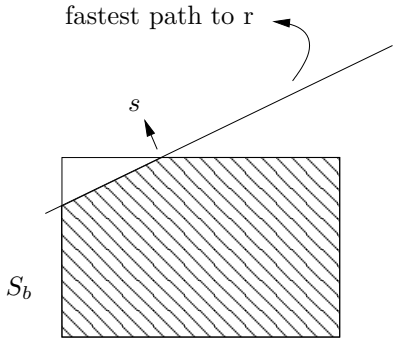


Figure 6.12: Illustration of flexibility.

6.4.2 Measurement errors and space-time prisms

A common strategy in error analysis is to model errors on observed values and analyse how these propagate onto derived values. Using the envelope from Section 6.2, we can do two things.

Firstly, since our sample regions are bounded sets by definition, the enve-

lope is the union of all possible space-time prisms with anchors in the sample regions.

Secondly, as is common in error analysis, we can compute a confidence region on the spatial and temporal component separately of each region, this is again a box-shaped region and compute the envelope with that confidence region as a new sample region. We note that we can not use a simultaneous confidence region for both space and time since those regions are usually elliptical in nature. This again returns the union of all space-time prisms with anchors in those confidence regions.

6.5 Conclusions and future work

Space-time prisms model uncertainty between sample points, which are usually considered to be perfect measurements. In this chapter we extended the notion of sample points to sample regions, which are bounded, sometimes disconnected, subsets of space-time wherein each point is a possible location, with its respective probability, where a moving object could have originated from or arrived in. This model allows us to model measurement errors, multiple possible simultaneous locations and even flexibility of a moving object.

We developed an algorithm that computes the envelope of all space-time prisms that have an anchor in these sample regions and we developed an algorithm that computes for any time-space point the probability with which a space-time prism, with anchors in these sample regions, contains that point.

We left out a complexity consideration. It is straightforward once you have the number of nodes obtained from the pre-computation, but obtaining that number however is not so trivial. This would rely on defining suitable heuristics that relate the width of temporal intervals to the number of possible distinct edges that can be travelled on, which in turn also relies on network density. But these are all aspects we left out in our simplified model and can be tackled in future work.

As for scalability, when treating a chain of prisms, i.e., a lifeline necklace, each prism can be computed separately and completely independent. Our algorithm is thus very scalable on large trajectory samples.

In the simplest case, when all distributions on space and time are uniform, the uncertain prism has a lot of symmetry. Further study is needed to exploit this symmetry and speed up the computation.

In this chapter we restricted our sample regions to box shapes, this can easily be extended to other shapes because the intersection we need to compute remains the same, i.e., the intersection of one or two half-spaces with the sample region.

One aspect that has not been studied in this paper is a measure to rank a space-time point's probability inside a single prism. Intuitively one can imagine that points near the edge of the prism are less likely that points on the interior, however, a suitable likelihood function to express this intuition has not been found. A logical step is then to figure out a way to combine both measures and interpret them in a sensible manner.

7

Conclusion

In this final chapter we reflect on our contributions and give a short introduction to some remaining problems. The quantity that expresses the amount of change over time is speed and that quantity keeps recurring throughout all chapters. Speed is indispensable in the study of moving object databases and is actually there all along in spatio-temporal databases, where positions are stored along with a time stamp. Even GPS-equipped devices store a computed speed value.

In Chapter 3, we determined the class of transformations of trajectories for which speed is invariant, we then extended this class to all transformations that map space-time prisms onto space-time prisms. We used this to identify a query language, $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$, on a spatio-temporal database with which we are able to express queries about moving objects and their speed. It turns out that this language also comes across as natural to express properties of space-time prisms. We prove the soundness and completeness for this language when it comes to $\mathcal{V}$-invariant queries. The challenge here is to do the same for road networks. The class of transformations will likely not change much but delivering a complete query language to express properties of trajectories and space-time prisms on road networks is not trivial. The main reason is that the underlying space is different. Movement on road networks is not only constrained to the network, the speed of a trajectory on a road network is also bound, on a per-segment basis. Distances on road networks is another obstacle to overcome. A road network is no longer a metric space and distances are not measured by a formula anymore, but by an algorithm which

depends on the networks itself. Nor can you call it a distance anymore since a distance between two points might not exist and the distance from A to B is usually not equal to the distance from B to A.

The interesting aspect of space-time prisms is that they, given an upper bound on a moving object's speed, capture all possible movement of this object. It is the subset of space-time of all spatio-temporal points where a moving object *could* have been. In this sense it captures the uncertainty of a moving object's position between measured time-stamped locations, i.e., sample points. A query of interest in this model is the *alibi query*. This query decides whether or not two moving objects *could* have met, in which case they have an alibi for not doing so. Answering this query in the space-time prism model comes down to deciding whether two space-time prisms intersect. Space-time prisms have an $\mathsf{FO}(+, \times, <, 0, 1)$-expression and the alibi query can then be expressed as a quantifier elimination problem. Since the first-order language over the real number allows quantifier elimination we know that a quantifier-free version of this expression exists and can be computed. The algorithms to do this [15, 30, 32] require a lot more computation power than we have available at this time. In Chapter 4, we first present a solution to the alibi query for movement in one dimension. Then we proceed to movement in two dimensions. We classified the intersection of two space-time prisms into three cases, all of which have, using a geometric argument, a quantifier-free expression. This results in a quantifier-free formula that can be used to effectively evaluate the alibi query. There are some challenges left here. Firstly, there is the question if such a formula exists for more the intersection of more than two space-time prisms. Secondly, to compute the spatial and temporal projection of this intersection remains unexplored, but interesting, territory.

Since most of our movement is constrained to road networks, a lot of the unconstrained space-time prism is wasted. However, a space-time prism on a road network is not merely the intersection of the network and the unconstrained prism, nor is it straightforwardly derived from one-dimensional space-time prisms. In Chapter 5, we adapt the concept of a space-time prism to a road network, moreover, we provide algorithms to compute its structure and developed software to visualise these prisms. Completing the results from the previous chapter, we also provide an algorithm and implementation to solve the alibi query on road networks.

In Chapter 6, we move on from sample points to sample regions. Until now, sample points were considered exact measurements. The cylinder model allowed measurement errors and had this advantage over the space-time prism model. We introduce sample regions on road networks, box shaped regions in space-time to replace the sample point and model measurement uncertainty

in space and time. To cope with different types of errors and to distinguish between measured and self-reported errors, we added probability functions to model the different scenarios. We study how these errors and their probabilities propagated through a space-time prism. We create an algorithm to compute the fraction of space-time prisms that cover any spatio-temporal point and implemented these to create visualisations. One challenge concerning these space-time prisms is to study the probability of a trajectory inside a prism.

8

Publications

The results presented in this thesis have been published in several papers. We list these publications here.

Chapter	Reference
3	[17, 19]
4	[16]
5	[18]
6	submitted

Bibliography

[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] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, December 1959.

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

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] Floris Geerts, Sofie Haesevoets, and Bart Kuijpers. First-order complete and computationally complete query languages for spatio-temporal databases. *ACM Trans. Comput. Log.*, 9(2), 2008.

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

[12] 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.

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

[14] E. Helly. über mengen konvexer körper mit gemeinschaftlichen punkten. *Jber. Deutsch. Math. Vereinig.*, 32:175–176, 1923.

[15] H. Hong. QEPCAD — quantifier elimination by partial cylindrical algebraic decomposition. 1990. http://www.cs.usna.edu/ qepcad/B/ QEPCAD.html.

[16] B. Kuijpers, R. Grimson, and W. Othman. An analytic solution to the alibi query in the space-time prisms model for moving object data. *International Journal of Geographical Information Science*, To Appear.

[17] 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.

[18] B. Kuijpers and W. Othman. Modelling uncertainty of moving objects on road networks via space-time prisms. *International Journal of Geographical Information Science*, To appear.

[19] B. Kuijpers and W. Othman. Trajectory databases: data models, uncertainty and complete query languages. *Journal of Computer and System Sciences*, To appear.

[20] H.J. Miller. Modeling accessibility using space-time prism concepts within geographical information systems. *International Journal of Geographical Information Systems*, 5:287–301, 1991.

[21] H.J. Miller. A measurement theory for time geography. *Geographical Analysis*, 37(1):17–45, 2005.

[22] H.J. Miller and Y. Wu. Gis software for measuring space-time accessibility in transportation planning and analysis. *Geoinformatica*, 4:141–159, 2000.

[23] Barrett O'Neill. *Elementary Differential Geometry*. Academic Press, www.apnet.com, 2^{nd} edition, 1997. 482 pages.

[24] W. Othman. Implementations of spatio-temporal algorithms. 2007. http://othmanw.submanifold.be.

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

[26] D. Pfoser and C. S. Jensen. Capturing the uncertainty of moving-object representations. In *Advances in Spatial Databases (SSD'99)*, volume 1651 of *Lecture Notes in Computer Science*, pages 111–132, 1999.

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

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

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

[30] Thomas Sturm. Redlog. 2007. http://www.algebra.fim.uni-passau.de/ redlog/.

[31] 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.

[32] Eric Weisstein. Mathematica. 2007. http://www.wolfram.com.

[33] O. Wolfson. Moving objects information management: The database challenge. In *Proceedings of the 5th Intl. Workshop Next Generation Information Technologies and Systems*, pages 75–89. Springer, 2002.

Samenvatting

Deze thesis raakt aan verschillende onderzoeksgebieden. We behandelen en combineren onderwerpen uit de Constraint Database Theory, Geographical Information Science (GIS) en zelfs toepassingen uit Time Geography. De gemeenschappelijke noemer van al deze gebieden zijn moving object databases (MODs), databases van bewegende objecten.

Tegenwoordig zijn meer en meer apparaten, zoals GSMs en navigatietoestellen, uitgerust met locatie bewuste technologie, oftewel location aware technology (LAT). Deze toestellen, hetzij op mensen, voertuigen of dieren, produceren trajecten. Er zijn twee soorten trajectdata. Ten eerste zijn er *trajecten*, oftewel krommen in het reële vlak $\mathbf{R}^2$ geparametrizeerd door de tijd. Ten tweede beschouwen we *traject samples*, dewelke vooral bekend zijn bij MODs, deze zijn eindige sequenties van tijd-tuimte punten (elementen van $\mathbf{R} \times \mathbf{R}^2$). Een trajectdatabase bevat dan een eindig aantal trajecten of traject samples die voorzien zijn van een label.

8.1 Snelheid en space-time prisms

De meest natuurlijke grootheid die beweging van een object beschrijft, is *snelheid*. Deze grootheid geeft per definitie de mate weer waarin een object zijn positie wijzigt in functie van de tijd. Daarom zijn snelheid en snelheidslimieten centraal in deze thesis.

Er zij verschillende manieren om een traject te reconstrueren van een traject sample, lineaire interpolatie is de meest populaire techniek om dit te doen (zie pagina 85 van [11]). Maar deze techniek veronderstelt dat het object tussen de sample punten zich in een rechte lijn beweegt en dit tegen de minimale snelheid nodig om het volgende punt op tijd te bereiken. Dit is realistisch als de sample punten frequent en regelmatig zijn. Het is echter realistischer te veronderstellen dat een object een bovengrens heeft op zijn snelheid, die ofwel fysiek bepaald is, ofwel bij wet is vastgelegd zoals op wegennetwerken. Met

zo'n bovengrenzen heeft men reeds een onzekerheidsmodel opgebouwd waarin men *space-time prisms* tussen opeenvolgende sample punten beschouwd. Enkele eigenschappen van dit model werden reeds enkele jaren geleden behandeld door de GIS gemeenschap, onder andere door Pfoser en Jensen [26], Egenhofer en Hornsby [4, 5] en Miller [21], maar deze space-time prisms waren reeds gekend in de time geography van Hägerstrand in de jaren 70 [13].

Als een object zich vrij kan bewegen in elke richting, dan is het space-time prism de doorsnede van twee kegels (eentje met als top die wijst in de richting van de tijd en eentje waarvan de top tegen de richting van de tijd wijst) in ruimte-tijd en bevat het alle mogelijke trajecten van bewegende objecten die zich van het ene punt naar het andere begeven met als enige beperking dat ze zich aan de snelheidslimiet houden. Egenhofer noemt een keten van zulke space-time prisms een *lifeline necklace* [4]. Figuur 8.1 illustreert het concept van deze space-time prisms en een lifeline necklace.

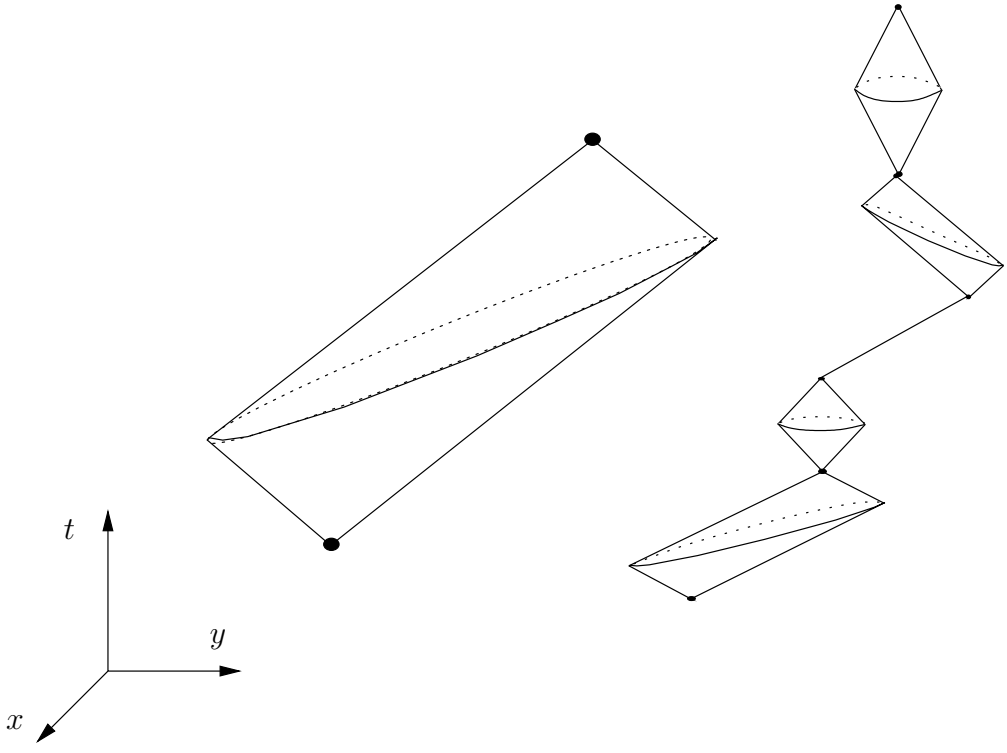


Figure 8.1: Een space-time prism (links) en een lifeline necklace (rechts).

8.2 Volledige querytalen

Snelheid is niet enkel van belang bij het opstellen van onzekerheidsmodellen. Bij het bevragen van traject data komen verschillende fysische grootheden ter sprake waarvan snelheid de meest belangrijke is. Geerts ontwikkelde een model waarin men de bwegingsvergelijkingen van een object direct op voorhanden heeft, in plaats van traject samples, hij veronderstelde ook dat de werkelijke snelheid altijd gekend was [9]. Als we vragen willen stellen over snelheid, dan is het ook belangrijk te weten welke transformaties van tijd-ruimte, voorgesteld door $\mathbf{R} \times \mathbf{R}^2$, de snelheid van een traject onveranderd laten. Deze groep van transformaties noemen we $\mathcal{V}$, en zijn samengesteld uit affiniteiten van de tijd met orthogonale transformaties van de ruimte en ruimtelijke homothetieën met dezelfde schalingsfactor als de temporele affiniteit en translaties. Geerts et al. [10], behandelen transformaties die de snelheid onveranderd laten, maar zij gaan uit van ruimtelijke transformaties die een functie zijn van enkel de tijd. De resultaten die we in deze thesis bekomen gelden voor algemene afleidbare transformaties van tijd-ruimte. In Hoofdstuk 3 tonen we aan dat de groep $\mathcal{V}$ ook de transformaties omvatten die space-time prisms invariant laten. Dus queries die met snelheid te maken hebben zijn invariant voor transformaties van $\mathcal{V}$, net zoals queries die met onzekerheid in termen van space-time prisms omgaan. Juist omdat queries die snelheid in acht nemen en queries die omgaan met onzekerheid door middel van space-time prisms, is het aangeraden een querytaal te gebruiken die invariant is onder transformaties van $\mathcal{V}$. Space-time prisms zijn al eerder in een onzekerheidscontext bestudeerd [4, 5, 21, 26], maar nog niet in de context van een querytaal.

Als uitgangspunt om trajectdatabases te bevragen, gaan we uit van een twee-ledige logica gebaseerd op de eerste orde logica over de reële getallen, in dewelke we beschikken over label variabelen en reële variabelen. Eerste orde logica over de reële getallen is reeds uitvoerig bestudeerd in de context van Constraint Databases [25]. Deze logica is expressief genoeg om eigenschappen van snelheid en space-time prisms uit te drukken. We merken op dat de $\mathcal{V}$-invariante queries een onbeslisbare klasse vormen. Bovendien tonen we aan dat deze taal bevat zit in een drie-ledige logica, dewelke label variabelen, tijd-ruimte puntvariabelen en snelheidsvariabelen (positieve reële getallen) bevat, en welke twee eenvoudige predicaten bevat: $\mathsf{Before}(p, q)$ en $\mathsf{minSpeed}(p, q, v)$. De eerste drukt uit dat voor twee tijd-ruimte punten p en q, tijd-component van p kleiner is dan die van q. De tweede drukt uit dat de minimale snelheid nodig om q vanuit p te bereiken gelijk is aan v. Deze logica laat veeltermongelijkheden toe op de snelheidsvariabelen. We tonen aan, door middel van deze twee conceptueel intuitieven predicaten, dat we alle $\mathcal{V}$-invariante eerste orde queries kan uitdrukken. Deze logica laat ons toe alle queries over snelheid van

trajecten en onzekerheid door middel van space-time prisms uit te drukken. We kunnen zelfs het predicaat $\mathsf{inBead}(r, p, q, v)$, dat uitdrukt dat r zich in het space-time prism van p en q met snelheidslimiet v bevindt, uitdrukken in deze taal.

We tonen ook aan dat een programmeertaal, gebaseerd op deze drie-ledige logica, in dewelke we relaties kunnen aanmaken en waarin we beschikken over een while-loop met eerste orde stop condities, volledig is voor de berekenbare $\mathcal{V}$-invariante queries op traject databases. De bewijzen van deze volledigheidsresultaten zijn geinspireerd door eerder werk over volledige talen op spatiale databases [12] en over spatio-temporele databases [10]. De taal die we hier voorstellen is veel meer gebruikers-gericht dan de taal voorgesteld door Geerts et al. [10], vooral omdat onze taal gebaseerd is op snelheid in plaats van meetkundige predicaten.

8.3 Kwantoreliminatie en de alibi query

Een interessante query die reeds bestudeerd werd door Egenhofer en Miller [4, 5, 21], is de *alibi query*. Deze booleaanse query vraagt of twee bewegende objecten, gegeven door traject samples en snelheidslimieten, elkaar fysiek ontmoet kunnen hebben. Het komt erop neer te kunnen beslissen of twee lifeline necklaces van space-time prisms van deze bewegende objecten een niet-lege doorsnede hebben of niet. Dit probleem heeft een praktische oplossing als we efficient kunnen nagaan of twee space-time prisms een niet-lege doorsnede hebben of niet.

Benaderende oplossingen zijn reeds voorgesteld [4], maar een exacte oplossing bestaat ook. We tonen aan dat de alibi query een uitdrukking heeft in het constraint database model door middel van een eerste orde constraint database query [19, 25]. Het is gekend dat eerste orde constraint queries effectief geevalueerd kunnen worden en dat er implementaties bestaan van kwantoreliminatie algoritmen voor eerste orde logica over de reële getallen om deze queries te evalueren, zie Hoofdstuk 2 van [25]. Experimenten met software zoals QEPCAD [15], RedLog [30] en Mathematica [32] op meerdere space-time prisms tonen aan dat het beslissen of twee concrete space-time prisms een doorsnede hebben kan berekend worden op gemiddeld 2 minuten (in Windows XP Pro, SP2, met een Intel Pentium M, 1.73GHz, 1GB RAM). Dit betekent dat, om deze query te evalueren op twee lifeline necklaces van bewegende objecten van elk 100 space-time prisms, ongeveer $100 \times 100 \times 2$ minuten zou duren als we alle paren van space-time prisms op intersectie zouden testen, wat ongeveer gelijk is aan twee weken. Een minder naieve methode bestaat erin eerst de tijdsintervallen op intersectie te testen, maar dan is de rekentijd

ongeveer gelijk aan $(100 + 100) \times 2$ minuten, oftewel ongeveer 7 uur. Beide tijden zijn niet aanvaardbaar vanuit een praktisch standpunt.

Een andere oplossing binnen het bereik van constraint databases, is een formule te vinden waarin de coordinaten van de traject samples en de snelheidslimieten als parameters voorkomen, en die uitdrukt dat twee space-time prisms een niet-lege doorsnede hebben. We noemen dit probleem de *parametrische alibi query*. We kunnen, althans in theorie, een kwantorvrije versie vinden door een blok van drie existentiele kwantoren te elimineren door middel van software zoals MATHEMATICA en QEPCAD, maar na enkele dagen runnen hebben we de berekeningen onderbroken zonder een oplossing. Het is algemeen bekend dat die algoritmes het bijzonder moeilijk hebben met hoger dimensionele problemen. Het voordeel van een kwantorvrije formule is dat de alibi query dan in constante tijd kan worden beantwoord. het beslissen of twee lifeline necklaces een niet-lege doorsnede hebben kan dan berekend worden in een tijd die evenredig is met de som van het aantal space-time prisms in elke necklace.

De hoofdbijdrage van Hoofdstuk 4 is een analytische oplossing voor de alibi query in isotrope twee-dimensionele ruimte, wat een oplossing is voor een probleem dat al sinds 2001 gesteld is. De kwantorvrije formule die we geven bevat evenwel wortels maar we geven eveneens aan hoe we deze wortels kunnen elimineren. De basis voor onze oplossing is een meetkundige beschrijving van de drie manieren waarom twee space-time prisms kunnen snijden. Deze drie gevallen kunnen dan vertaald worden in een efficiente formule waarmee we de alibi query op twee necklaces van elk 100 space-time prisms kunnen berekenen in minder dan een minuut. Dit geeft ons een praktische oplossing voor de alibi query.

8.4 Space-time prisms op wegennetwerken

In Hoofdstuk 5, bestuderen we bewegende objecten en space-time prisms op *wegennetwerken*. Eerder werden al aanpassingen van het space-time prism model voor wegennetwerken voorgesteld door Miller [20, 22], alwaar ze concepten zoals *network time prism* en *potential path tree* introduceerden. Het eerste is de set van alle mogelijk bezochte segmenten in het wegennetwerk en het tweede is een deelboom van het eerste.

We beschouwen wegennetwerken als een graaf inbedding in $\mathbf{R}^2$ waar de bogen rechte lijnen zijn tussen de vertices. Alle bogen hebben een strikt positieve snelheidslimiet en gewicht, de *time span*, die gelijk is aan de minimale tijd die nodig is om de boog te overbruggen gegeven de snelheidslimiet.

Een bewegend object is gegeven door een lijst van time-space punten

$(\mathsf{t}_i, \mathsf{x}_i, \mathsf{y}_i)$ met $i = 1, \ldots, N$, en $(\mathsf{x}_i, \mathsf{y}_i)$ op het wegennetwerk. Het is mogelijk dat een snelheidslimiet enkel afhangt van het object, maar in het algemeen werken we met snelheidslimieten die van boog tot boog kunnen varieren.

Het eerste probleem dat we aanpakken is het berekenen en visualizeren van een space-time prism tussen twee sample punten op een wegennetwerk, gegeven de snelheidslimieten op het wegennetwerk, alsook de visualisatie van de spatiale en temporele projecties van zo'n space-time prism. De hierboven vermelde time span is de sleutel tot de berekening van space-time prisms op wegennetwerken. Kortste pad-lengte op een wegennetwerk noemen we ook *road network time* en is de kortst mogelijke tijd om van een punt in het wegennetwerk naar een ander te gaan. Als de snelheidslimieten uniforn zijn dan is deze afstand proportioneel met de traditionele kortste pad-lengte. Onze algoritmes zijn onder meer gebaseerd op het algoritme van Dijkstra [3]. De complexiteit van ons algoritme is kwadratisch in een deel van het aantal vertices in het wegennetwerk. De output is een polygonale representatie van het space-time prism in ruimte en tijd. Ter illustratie hebben we deze algoritmen in MATHEMATICA [24] geimplementeerd.

Gebruikmakend van deze polygonale representatie hebben we ook algoritmen ontwikkeld om de *alibi query* op wegennetwerken te berekenen. We beginnen met een algoritme om te beslissen of twee space-time prisms van bewegende objecten op een rechte elkaar snijden of niet. Dit levert een eerste orde formule die kan geevalueerd worden in constante tijd. Dan gebruiken we deze oplossing om de alibi query op een wegennetwerk te beantwoorden. De evaluatie van deze query blijkt eenvoudig en snel op een wegennetwerk, gegeven dat we reeds de space-time prisms hebben berekend. Ook berekenen we de spatiale en temporele projecties van de doorsnede.

8.5 Space-time prisms en onzekere ankerpunten

Tot nu werden de sample punten, ook wel ankerpunten van space-time prisms genoemd, als exact gemeten punten beschouwd. Meetfouten worden daarmee genegeerd. In Hoofdstuk 6 laten we deze veronderstelling dat ankerpunten effectief punten zijn, vallen en we veralgemenen ze naar onzekerheidsregio's.

In praktijk zijn gemeten punten nooit exact. Ze kunnen evenwel zeer nauwkeurig zijn als ze gemeten zijn door apparaten met een GPS ontvanger (zelfs dan is de nauwkeurigheid tot op een meter of minder), maar ze kunnen ook heel onnauwkeurig zijn, zoals locatiedata op basis van GSM-masten. In dit laatste geval kan de onnauwkeurigheid zelfs enkele kilometers bedragen, en als we de doorsnede nemen met een wegennetwerk worden deze regio's zelfs onsamenhangend. Een derde voorbeeld dat de tekortkomingen blootstelt komt

voor wanneer mensen worden gevraagd om een dagboek van hun activiteiten bij te houden, bijvoorbeeld, “Ik vertrok naar huis tussen 5u en 5u30” of “Ik was in de buurt van de supermarkt omstreeks 8u30”. In beide gevallen is er onzekerheid in de ruimte en de tijd. Het laatste geval toont zelfs aan dat deze regio’s niet vloeiend in de tijd hoeven verbonden te zijn. Als de supermarkt pas open is vanaf 8u30 dan kan men daar niet geweest zijn voor 8u30.

De onzekerheidsregio’s modelleren alle bovenstaande gevallen. Dit zijn rechthoekvormige regio’s in tijd-ruimte bovenop een wegennetwerk. Die regio’s hoeven niet samenhangend te zijn, noch vloeiend in elkaar over te gaan. Om verschillende scenario’s te modelleren laten we ook probabiliteitsfuncties toe op deze regio’s.

In Hoofdstuk 6 introduceren we onzekerheidsregio’s en de omhullende space-time prism. We definieren ook drie grootheden die we aan elk tijd-ruimte punt toekennen, een eerste grootheid geeft de waarschijnlijkheid weer ten opzichte van de start onzekerheidsregio’s, een tweede grootheid doet hetzelfde ten opzichte van de aankomst onzekerheodsregio’s en de derde grootheid combineert de vorige twee. We geven ook nog algoritmes om de grootheden en geintroduceerde begrippen te berekenen en visualiseren.