ulm university universität
uulm

Support of Relationships Among Moving Objects on Networks

Markus Kalb, Claudia Dittrich, Peter Dadam

Ulmer Informatik-Berichte

Nr. 2008-05
Februar 2008

Support of Relationships Among Moving Objects on Networks*

Markus Kalb, Claudia Dittrich, Peter Dadam

Institute of Databases and Information Systems
Faculty of Engineering Science and Computer Sciences
University Ulm
James Franck Ring, 89081 Ulm
phone: +49 731 50-24126
{markus.kalb,peter.dadam}@uni-ulm.de

Abstract. Efficient support of moving objects on networks is receiving increasing attention. An interesting issue is to analyze the interactions of objects within a network. In order to support queries related to the dynamic behaviour of two or more moving objects in conjunction, topological information must be maintained which relates the moving objects to the network as well as the moving objects among another. As typical application areas tend to generate large data volumes, an adequate storage representation and the resulting effort to evaluate such topological relationships are crucial in practice. To face that challenge, we propose a new representation model for moving objects, which enables a rather compact representation of both, information on the network and information on the moving objects on this network. This, in turn, allows to efficiently evaluate respective queries.

1 Introduction

The support of spatio-temporal information in database and information systems is an important requirement in many application areas. Managing moving objects which change their positions in course of time has received increasing interest. Typical objects for this are, for example, cars, airplanes, or mobile phone users.

A specific challenge represents the interaction of different moving objects among each other, especially if a large quantity of objects is involved. Of particular interest in this context are objects which move on predictable routes on an underlying network. Examples are vehicles (cars, trucks, trains) which use given roads or rails to reach their destinations, satellites on a given orbit, or airplanes on pre-determined air-routes (air corridors). The network can be understood as a shared resource which restricts the possible movements of the objects and defines speed limits and maximal capacities of sections. Typical interesting questions are the determination of positions of objects, expected arrival times,

* This work is partially supported by the German Research Community within the Project "Periodicity in Information Systems"

prognosis of possible collisions, or traffic jams on routes. In addition, questions of the following kind are interesting:

- "Will bus line A reach stop X in time, so that the passengers in bus line B can change?"
- "In which section will the hazardous freight transport A pass the passenger train B?"
- "Where can express train A overtake the regional train B most favorably?"
- "Problem with a cargo truck: which trucks of the same company are on sections before or following it?"

The literature related to moving objects concentrated so far mainly on interactions of moving objects with static objects (e.g. [20,11,17]). For example, "When does object A reach a certain point?" or "When does object B transit area X?". The adequate treatment of relationships among two or more moving objects, as outlined above, have received little attention, so far. These relationships pose a great challenge regarding both the adequate representation of the movement information (data volume!) and the efficient evaluation of queries. Especially, if one keeps in mind that in real life scenarios typically a large number of objects has to be considered.

The paper is organized as follows. Section 2 describes the state of the art. In Section 3, the application scenario is described in detail as well as fundamental terms and definitions are introduced. In Section 4 movement-dependent relationships are introduced, which forms the basis for the representation and evaluation strategies. In Section 5 the logical representation model is presented and the evaluation of the relationships is described. In Section 6 implementation issues are discussed and Section 7 concludes the paper with summarizing remarks.

2 Related Work

The majority of approaches dealing with relationships only consider relationships among static and dynamic data or objects moving absolutely free in space. For example, [21,19] introduce so-called trajectory-based queries which evaluate relationships among trajectories and a static area. Similar relationships with respect to objects on a network have been discussed in [7,11,12]. Recent approaches like [2,17] additionally address the problem of efficiently finding common sections among moving objects on a network or calculating the numbers of objects within a specific section at a specific time [4].

Erwig and Schneider exploit in [9] the work on topological predicates for spatial objects (e.g. [8]) done so far and add the temporal aspect. They consider spatio-temporal objects as function from time into space. The topological relations among moving objects are specified by the so called spatio-temporal predicates. The predicates hold over time intervals (period predicates) or at time points (instant predicates) and their composition builds up a "'development"'. Erwig and Schneider limit the spatio-temporal predicates to some elementary

predicates, the so called canonical collection of spatio-temporal predicates, from which more complex ones can be composed. In the case of two moving points, which are relevant in our study, they only specify the two elementary spatio-temporal predicates *Disjoint* and *Meet*. To formulate the relevant relationships in our study we have to build up a sequence of these two spatio-temporal predicates and other spatial predicates. However, the spatio-temporal predicates do not make a distinction between relationships among objects moving on the same itinerary and the relation among objects moving on different itineraries. But, this disitinction is critical in our work. Such relationships among two or more moving objects on a network have not been treated well in the literature so far.

The efficient evaluation of relationships among moving objects requires an adequate representation of their trajectories. The majority of existing approaches for an efficient evaluation and representation of moving objects and their trajectories are mostly situated in the context of index and access structures. Typically, the trajectory of an object is modeled as a linear function of time (e.g. [3,14,21,15,18,22]). Thus, the trajectory or the graph of the functions becomes a line or polyline within the euclidean space, i.e. within the space-time domain ($\mathbb{R}^2 \times T$) or ($\mathbb{R}^3 \times T$). By doing so, existing geometrical algorithms for polylines as well as related indexing techniques can be used to evaluate queries.

However, most of the index related approaches focus on objects moving absolutely free in the geographical space, without any limiting underlying network. They do not regard the specific characteristics of moving objects on networks. Each object is treated individually, i.e. the spatial information of their itinerary is maintained for every object separately, even if they use the same itinerary. With respect to the application scenario, this leads to a high degree of redundantly stored information and thus to high volumes of data.

One class of approaches represents or indexes trajectories in the original (euclidean) space-time domain. The trajectories or rather their multidimensional polyline are approximated by (multidimensional) minimal bounding rectangles (MBR), like in classical R-trees. Typical instances are for example the partial-persistent R-tree (PPR-tree) [15], spatio-temporal R-tree (STR-tree) and the trajectory-bundled tree (TB-tree) [21]. With respect to moving objects such approaches have two significant problems. Firstly, due to the approximation of an polyline by MBRs a high degree of "dead space" is generated. It means, the MBR approximation has a much larger extend in space than the polyline. This effect is highly increasing in case of long time movements. Secondly, moving objects on a network move along predefined sections. Due to frequently used sections, the path of objects do overlap. This in turn, leads to a large amount of overlapping MBRs. Both problem results in a low degree of selectivity in such index structures. The selectivity problem and the high data volumes caused by the redundancy problem, prohibit the application of such approaches in scenarios where a large number of objects has to be considered.

The problem of "dead space" is treated in some approaches by mapping the original trajectories into a parametrized space [14,3,23,16]. Depending on the problem setting different motion parameters, like velocity, location, or start and

end times can be used. The general idea is that the original trajectory or rather every section of its multidimensional polyline becomes a multidimensional point in a parametrized space. This in turn makes it easier to use MBRs without having the dead space problem. Even with the (partial) loss of spatial or temporal information within such alternative spaces it was shown, that range queries like *"select all objects which are within rectangle x at a specific point in time"* can be efficiently supported. However, the redundancy problem as well as the problem of frequently used sections remain the same.

To better represent movements of objects in networks dedicated storage and index structures have been proposed which utilize properties of the underlying network to allow a more compact representation. One class of approaches splits the trajectories in their temporal and spatial characteristics (e.g. [10,20,1]). They use two separate structures to index both the network and the motion data of the moving objects. These index structures are interconnected. However, an efficient access is only provided along one dimension, i.e. even the temporal or the spatial dimension. That is, the indexed dimensions are not treated symmetrically. By doing so, they are not very well suited to support the full spectrum of relationships among moving objects (see Sect. 4). Another class of approaches uses a grid-based storage or index structures (e.g. [4,25]). Instead of the extend of an object, the domain space of the objects is approximated and simple methods are provided to locate the data. It was shown that such a partitioning approach is more appropriate in the contex of moving objects on networks [4]. Most closely-related to our work is the space-time partitioning approach introduced in [4,5,7,6]. The authors use the same representation of polylines to reduce the spatial information. However, they solely concentrate on the interplay of capacities of network sections and their effect on objects moving on it (traffic jams, speed adjustment). Relationships among objects or their evaluation have not been regarded.

3 Application Scenario and Basic Definitions

As already mentioned in Sect. 1 we focus on objects like cars, trucks, or airplanes moving on underlying networks, like roads or air-routes. In this paper, we are primarily interested in answering questions concerning movement-dependent relationships among such objects. Therefore, a rather simple network model is sufficient for our purposes (as opposed to, e.g. [11], where one is interested also in detailed information on the network itself). In the following, we use a simple graph to represent a *network*, where edges represent sections (streets, rails, air-routes, etc.) and points represent junctions. An example for such a network is shown in Fig. 1, where the network represented by the solid lines and the routes taken by the objects are represented as dashed lines. Formally, a network is defined as follows[1]:

[1] The definition 1 can be easily extend to the three-dimensional geographical space domain.

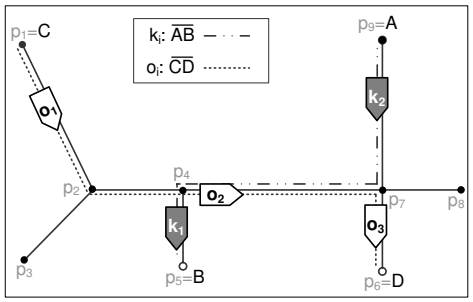


Fig. 1. Moving objects and their itinerary within a network

Definition 1. ***(Network)*** *A tuple $N = (P, S)$ is called a network,* iff *P is the set of points $p \in \mathbb{R}^2$ from the spatial domain, S is the set of sections $(p_i, p_j) \in P \times P$, such that $N = (P, S)$ is a graph.*

A network restricts solely the possible movements of objects. However, the specific route of a moving object in the network is definied by its *itinerary*. An itinerary defines a valid trail along the sections of the network from a start point to an end point. For example, Fig. 1 shows an itinerary from point A to B ($\overline{AB}$) defined by trail (p_9, p_7, p_4, p_5) and a second itinerary from C to D ($\overline{CD}$) along the points $(p_1, p_2, p_4, p_7, p_6)$. Furthermore, the objects o_1, o_2, o_3 move along itinerary $\overline{CD}$ and the objects k_1, k_2 move along itinerary $\overline{AB}$. Formally, an itinerary can be defined as follows[2]:

Definition 2. ***(Itinerary)*** *An itinerary I valid for a network N is the list of points $I_N = (p_0, ..., p_n)$, iff: I_N is an acyclic trail in N.*

Objects having the same itinerary and therefore use the same way in space, can have individual starting times and velocities. This in turn, leads to different arrival times for every object at the junction points of its trail. This time-dependent trace of positions of an object, is named its *trajectory*. In the following formal definition of the trajectory of moving objects, we use the continuous time domain $\mathbb{R}$. The time t_0 corresponds to the starting point of the object.

Definition 3. ***(Trajectory)*** *T^I is called the trajectory of a moving object on itinerary I_N, iff: $T^I = ((p_0, t_0), (p_1, t_1), ..., (p_n, t_n))$ with $(p_i, t_i) \in I_N \times \mathbb{R}$, $i = 0, .., n$.*

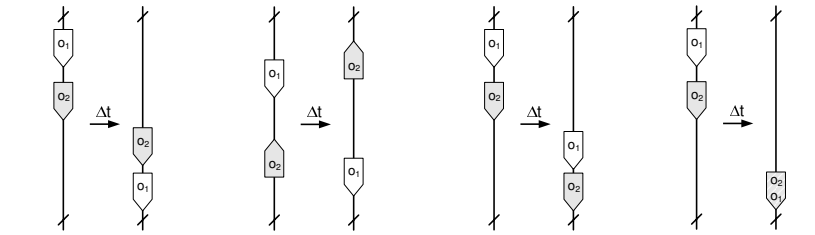
It depends on the application scenario whether a trajectory represents past or future movements.

[2] We do not consider cyclic itineraries or itineraries containing loops. This is part of our future work.

4 Relationships among Objects in a Network

An important issue is to evaluate the interaction among objects moving on a network. Looking at the figure above, an interesting question is *"will object o_2 pass object k_2"* (on their shared section p_4, p_7). Such relationships state the behavior of two moving objects on a specific place (Δs) during a specific time interval (Δt). Related to a two-dimensional scenario, Fig. 2 shows four relevant relationships among objects using the same section on the network. The cone end shows the movement direction of an object.

The semantics of the first and second relationships, i.e. *overtake* and *pass,* are quite obvious. The relationship *follows* indicates that object o_1 is always "behind" o_2 on this section (although the distance may change). In contrast, the relationship *meet* states that both objects coincide with each other on that section.



(a) o_1 *overtake* o_2 (b) o_1 *pass* o_2 (c) o_1 *follows* o_2 (d) o_1 *meet* o_2

Fig. 2. Relationships among objects on the same section Δs

Regarding objects on different but intersecting sections, the relationship *cross* represent the only meaningful relationship (in a two-dimensional domain). This relationship indicates that the two objects get together at the intersection point of their sections, as shown in Fig. 3. This relationship can also hold, if two sections only touch each other at a single point (see sections (p_3, p_2) and (p_1, p_2) in Fig. 1).

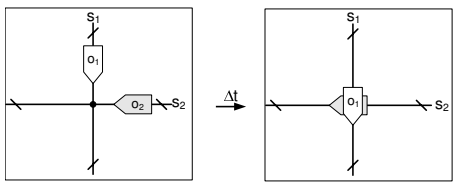


Fig. 3. Relation *cross* among objects on different sections

In general, among objects moving on the same itineraries several of this relationships can be valid during their runs. In Fig. 1, for example, the objects o_1 and o_2 show a *follows* relationship, but may also satisfy *meet* and *overtake* at a later point in time. However, they can never satisfy the relationships *pass* and *cross* (because of the assumption that trails are acyclic, see Def. 3).

Objects moving on different itineraries with only some points (but no sections) in common, can only satisfy the *cross* relationship. If, in addition, their itineraries have common sections then all these relationships may be valid. In Fig. 1 the objects using itineraries $\overline{AB}$ and $\overline{CD}$ can potentially satisfy the relationship *pass* on the shared section p_4, p_7 or satisfy the relationship *cross* at their junction points p_4 and p_7.

To determine whether these relationships hold (and if, at which point in time and space) is rather complicated to determine if objects can freely move in space. Fortunately, when considering objects moving on a given network, the "potential places and sections" where a certain relationship among two objects might be satisfied (called *correlation sections* in the sequel) are only depending on the itineraries and can be determined a priori. This aspect is specifically interesting for objects moving on different itineraries. For example: Whether object o_i and k_i will satisfy the relationship *cross* can only be determined at "run time". However, if the relationship is satisfied then it can only happen at point p_4 and this point can determined a priori. Analogously, the relationship *pass* can only be satisfied on section (p_4, p_7)[3]. Formally, *correlation sections* are definied as follows:

Definition 4. ***(Correlation section)*** *A correlation section between two itineraries with regard to a relationship $\otimes$ is the tuple $c_{\otimes}(I_1, I_2) = (p_i^1, p_{i+n}^1)$, with $I_1' = (p_i^1, ..., p_{i+n}^1) \subseteq I_1$, $I_2' = (p_j^2, ..., p_{j+n}^2) \subseteq I_2$ and $n, i, j \in \mathbb{N}$, such that:*

1. $\otimes \in \{meet, follows, overtake\} \Leftrightarrow \exists n \geq 1 : p_i^1 = p_j^2, p_{i+n}^1 = p_{j+n}^2$
2. $\otimes = pass \Leftrightarrow \exists n \geq 1 : \; p_i^1 = p_{j+n}^2 \wedge p_{i+n}^1 = p_j^2$
3. $\otimes = cross \Leftrightarrow \exists n = 0 : p_i^1 = p_j^2 \wedge p_{i+n}^1 = p_{j+n}^2 \wedge (p_{i-1}^1 \neq p_{j-1}^2 \wedge p_{i+1}^1 \neq p_{j+1}^2)$

A correlation section for the relationships *meet*, *follows*, *overtake* only exists, if both itineraries contain a common section. However, these relationships can only hold among objects moving in the same direction. But the direction of an moving object is also predefined by its itinerary, i.e. by their sections and the order of points. This is formally defined in the first item above, i.e. a correlation points correspond to the start point and end point of the sequence of points contained in both itineraries. This guarantees the same direction of the movement. The opposite holds for the relationship *pass,* which can only be satisfied if both objects are moving in opposite directions. Here, the common section of a corresponding correlation point must have a different direction, i.e. the order in the sequence of points must be opposite. In case of the *cross* relationship the section becomes a single point, i.e. an intersection or touching point of both

[3] Of course, an itinerary may change dynamically as well, for example, to avoid traffic jams. However, to simplify matters we do not consider this case and handle a changed itinerary simply as a new one.

itineraries. Two itineraries intersect or at least touch each other if they have points in common.

It is interesting to note that the determination of such correlation sections is completely based on the evaluation of the points given by the itineraries. This means, neither complex spatial representations nor respective "spatial algorithms" are needed.

Whether a certain relationship is satisfied between two moving objects, depends on their dynamic behaviour, that is if they are at the same place at the same point in time. To compute this efficiently an adequate representation of moving objects is required.

5 Evaluation of Relationships

The efficient evaluation of relationships among moving objects requires an adequate representation of their trajectories. As already mentioned in Sect. 2, in our application scenario the movement of all object is limited by the underlying network. This fact can be utilized to represent the spatial information in a much more compact way. As the objects of interest move along the same itinerary can commonly use the same spatial informations, i.e. most of this information needs to be stored only once for these objects. Only the velocity and the starting point in time are varying.

5.1 *KP* Representation of Trajectories

The "trick" to reduce the spatial information for objects moving on the same itinerary is to use an alternative reference system for the positions of objects. Such an alternative system is the *kilometer-post reference system* of positions (or 1,5-dimensional space; [24,5,13]). This reference system measures each position of an object as the covered distance from the starting point along its itinerary.

Let the function $dist(p_1, p_2)$ calculate the euclidean distance between two points $p_1, p_2 \in \mathbb{R}^2$. Formally, the transformation of a position (point) given in absolute coordinates into the kilometer-post reference system (KPRS) can then be defined as follows:

Definition 5. ***(Point representation in the KPRS)*** *The kilometer-post representation of a point $p \in \mathbb{R}^2$ is the covered distance $d \in D \equiv \mathbb{R}$ along the polyline pl, measured from the start point p_0,* iff*: pl is the list of points $(p_0, ..., p_n)$ with $p_i \in \mathbb{R}^2, i = 0, ..., n$ and point p is situated on the polyline pl in the section (p_j, p_{j+1}) with $j = 0, .., n-1$. PL is the domain of polylines,* such that*, the covered distance $d := dist(p_j, p) + \sum_{k=j}^{1} dist(p_{k-1}, p_k)$.*

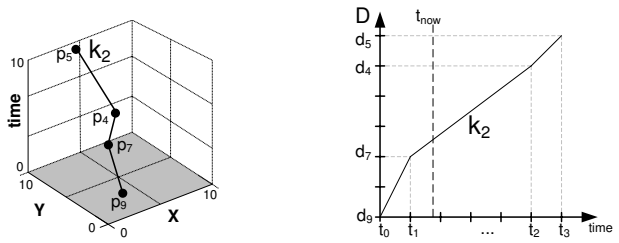
That is, the kilometer-post representation d of a point p is simply the euclidian distance to the starting point p_i of its section (p_i, p_{i+1}) plus the accumulated section lenghts back to the starting point p_0. This means, the position of an object, i.e. it two- or three-dimensional coordinates, is represented in the kilometer-post

reference system by a single real value. By doing so, the original two-(or three) dimensional space domain is reduced to a one-dimensional real value domain D.

Using the kilometer-post reference system within trajectories, i.e. for their spatial coordinates, and using again the continuous time domain $\mathbb{R}$, we can define the KP representation of trajectories as follows:

Definition 6. *(KP **representation**) kpT^I is called the KP representation of the trajectory T^I,* iff: *$kpT^I = ((d_0, t_0), ..., (d_n, t_n))$ is the list of pairs $(d_i, t_i) \in D \times \mathbb{R} \equiv \mathbb{R}^2$ and $i = 0, ..., n$,* such that, *$\forall (p_i, t_i) \in T^I : \exists (d_i, t_i) \in kpT^I$.*

Similiar to approaches based on a space-time domain (e.g., [4,7]) we also interpret the list of d values of the KP representation of an trajectory as a polyline in the distance-time domain. Using this approach, the multi-dimensional polyline of the trajectory of object k_2 of Fig. 1 as illustrated in Fig. 4a) is reduced to the two-dimensional polyline as shown in Fig.4b). The various gradients of the single lines correspond to the varying velocities of the object along its itinerary.



(a) multi-dimensional representation

(b) KP representation

Fig. 4. Alternatives to represent trajectories of moving objects

However, reducing spatial information in this way also means to lose or "hide" information. The question is whether this has a negative impact on the computation of relationships. This issue is discussed in the next sections. Two cases must be discriminated here: to determine toplogical relationships among objects on the same itinerary and such among objects on different itineraries.

5.2 Relationships Among Objects on the Same Itinerary

As already shown before (see Sect. 4), only the relationships *overtake, follows,* and *meets* can be satisfied among objects on the same itinerary.

Consider the KP representation of two objects, with different starting points in time (see Fig. 5). In case that their representations have one point $c = (d, t) \in D \times \mathbb{R}$ in common only means, in general, that both objects have covered the same distance d at the same time t. However, if both KP representations refer to

the same itinerary, the common point c corresponds also to the identical spatial position at this point in time.

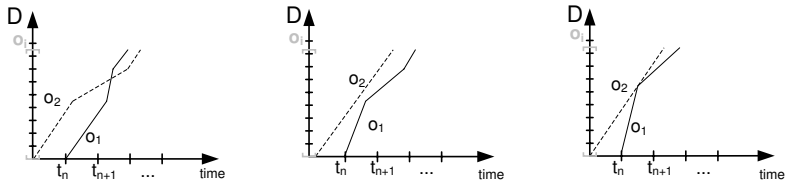
Thus, the geometrical relationships between two KP representations of objects following the same itinerary directly correspond to the relationships. For example, if two KP representations intersect, this directly corresponds to the relationship *overtake*. The intersection point states that both objects are at the same place (in space) at the same point in time. In addition, one object has a higher velocity (higher gradient of the polyline) and thus overtakes the other object.

In the following we assume an auxillary function $intersect(pl_1, pl_2)$, which determines the set of intersection points of two polylines pl_1 and pl_2. Formally, the relationship *overtake* based on a KP representation of two objects can then be defined as follows:

Definition 7. ***(Overtake)*** *Given the KP representation kpT_1, kpT_2 of the two trajectories T_1^I, T_2^I, then the function overtake : $M^T \times M^T \times \mapsto$ Boolean with M^T naming the set of all trajectories, determines whether two corresponding objects overtake each other:*

$$overtake(T_1^I, T_2^I) := \begin{cases} true & : intersect(kpT_1, kpT_2) \neq \emptyset \\ false & : else \end{cases}$$

In the definition above and in the rest of the paper the direction of a relationship is not relevant, i.e. $overtake(T_1^I, T_2^I) \equiv overtake(T_2^I, T_1^I)$. In a similiar fashion we can define functions for the relationships $follows$ and *meet* as illustrated in Fig. 5.



(a) o_1 *overtake* o_2 (b) o_1 *follows* o_2 (c) o_1 *meet* o_2

Fig. 5. Relations between relationships and geometrical relationships of the KP representations of two objects o_1, o_2 moving on the same itinerary

5.3 Relationships Among Objects on Different Itineraries

Moving objects on different itineraries can satisfy all the relationships mentioned in Sect. 4, i.e. *overtake*, $follows$,*meet*, *cross*, and *pass*. If the KP representations of two objects on different itineraries have one point in common, it also means (like in the previous section), that both objects have covered the same distance at the same time. But when considering objects which move on different itineraries

a common point does no longer mean that both objects are at this point in time at the same place. Instead, within the spatial domain the objects can be quite far away from each other and we can not conclude whether any of the relationships are satisfied.

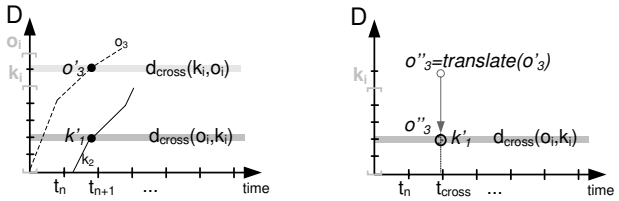
However, as explained in Sect. 4, we can determine the *correlation sections* (see Def. 4) which identify those sections on two different itineraries where these relationships among their objects can be satisfied. In the following we show how correlation sections can be used to adapt the KP representation such that the geometrical relationships of KP representations again correspond to relationships.

Like absolute coordinates of object positions, correlation sections can also be transformed into the kilometer-post representation. By doing so, the transformed correlation section identify intervals in the distance domain where common sections of two itineraries are located. However, as starting points of sections in the distance domain describe the relative distance to the origin of their itinerary, the transformation may lead to different interval values for identical sections in the space domain. In order to make the KP representation within this intervals comparable we have to adjust these intervals appropriately. This can be done by shifting one interval to the other such that their starting points match. Related to the KP representation this means that we shift it accordingly along the D axis (see Fig. 6). After this shift the geometrical relationships in the KP representations correspond again to relationships among objects in the space-time domain.

Let $\otimes \in \{overtake, follows, meet, cross, pass\}$ again denote the possible relationships and $c_{\otimes}(I_1, I_2)$ a correlation section of two itineraries (see Def. 4). Denote $d_{\otimes}(I_1, I_2)$ the respective transformation of the correlation section $c_{\otimes}(I_1, I_2)$ with respect to itinerary I_1. Accordingly, $d_{\otimes}(I_2, I_1)$ denotes the respective transformation of the correlation section $c_{\otimes}(I_1, I_2)$ with respect to itinerary I_2. Then the transformation of the correlation section $c_{\otimes}(I_1, I_2) = (p_i, p_j)$ leads to the intervals $d_{\otimes}(I_1, I_2) = (d_i, d_j)$, $d_i, d_j \in D$ and $d_{\otimes}(I_2, I_1) = (d'_i, d'_j)$ with $d'_i, d'_j \in D$. The factor of their translation is given by the vector $\overrightarrow{v} = (0,\ (d'_i - d_i))\,, \quad \overrightarrow{v} \in \mathbb{R} \times D$. The value of the temporal component of $\overrightarrow{v}$ is set to zero, because the translation only affects the distance-dimension.

In the definition below we assume two auxillary functions $clipping : PL \times R \times R \mapsto PL$ and $translation : PL \times \mathbb{R}^2 \mapsto PL$. The function $clipping(pl, d_s, d_e) = pl'$ clips a polyline pl regarding to an interval (d_s, d_e). The second function $translation(pl, \overrightarrow{v}) = pl'$ translates a polyline pl with respect to a vector $\overrightarrow{v}$.

Definition 8. ***(Cross)*** Given the KP representation kpT_1, kpT_2 of two trajectories $T_1^{I_1}$, $T_2^{I_2}$, then the function $cross : M^{T^I} \times M^{T^I} \times \mapsto Boolean$ determines whether two corresponding objects satisfy the relationship *cross, iff:* $d(I_2, I_1) = (d_1, d_2)$ *and* $d(I_1, I_2) = (d'_1, d'_2)$ *are the transformations of the correlation section* $c_{cross}(I_1, I_2) = (p_1, p_2)$, $\overrightarrow{v} = (0,\ (d'_1 d_1))$ *is the translation vector, and* $kpT_{ct} = translate(clipping(kpT_2, d_1, d_2), \overrightarrow{v})$ *is the clipped and translated KP representation of kpT_2such that:*



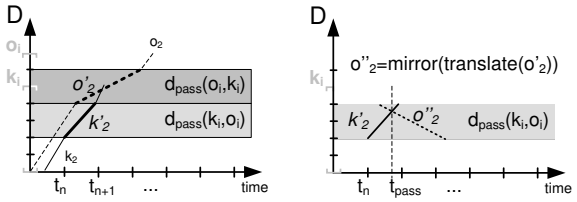
(a) Correlation sections of itineraries o_i and k_i (b) o_3 *cross* k_1

Fig. 6. Relation between relationship *cross* and geometrical relationship of the KP representations of two objects o_3, k_1 on different itineraries

$$cross(T_1^{I_1}, T_2^{I_2}) := \begin{cases} true & : intersect(kpT_1, kpT_{ct}) \neq \emptyset \\ false & : else \end{cases}$$

Figure 6 illustrates the relationship *cross*. In a similar fashion, the relationship *overtake*, *follows*, and *meet* can be defined.

The evaluation of the relationship *pass* additionally needs a *mirroring* of one of the two KP representations. The reason is, that the relationship *pass* (see Def. 4) describes that two objects move in opposite directions on the same section. In the distance-time domain the section between two points p_1 and p_2 is defined as (p_1, p_2) if the object is moving from p_1 to p_2, and it is defined as (p_2, p_1) if it moves into the opposite direction. The functions described above to evaluate the toplogical relationships are based on the KP representation which requires the identical order of points in trajectories in order to make them comparable (see Fig. 7 below).



(a) Correlation sections of itineraries o_i and k_i (b) k_2 *pass* o_2

Fig. 7. Relation between relationship *pass* and geometrical relationship of the KP representations of two objects o_2, k_2 on different itineraries

6 Evaluating Relationships in Large Data Sets

Evaluating relationships by sequentially scanning large sets of moving objects is too expensive, in general. Instead, suitable access and index structures for moving objects, i.e. of their KP-representations, are required. However, two significant properties of the $\mathbb{R} \times D$ domain must be considered here. Firstly, the objects to be indexed are diagonal lines or polylines. Secondly, geometrical relationships among such lines do not always correspond to relationships and vice versa. Additionally, the temporal dimension of the $\mathbb{R} \times D$ domain must be considered as continuously progressive dimension.

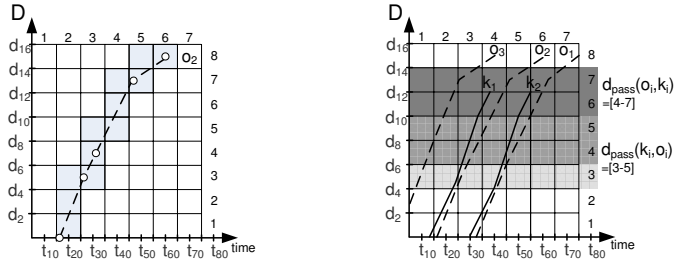
As already mentioned above, using minimal bounding rectangles to approximate (diagonal) lines, as it is done in R-Tress and related approaches, does not lead to satisfying results. Beside this, objects are clustered by their closeness, i.e. closely related objects are stored closely, as well. In the case of the $D \times \mathbb{R}$ domain this way of clustering does not always make sense semantically. Furthermore, due to the continuously progressive temporal dimension, constant restructuring of MBRs would be necessary.

As already shown by [4] using a partitioning approach, like a grid-based representation, is more suitable in the $D \times \mathbb{R}$ domain. In the next section we introduce a grid-based representation of trajectories and present algorithms for evaluating relationships.

6.1 Grid-based Representation of Trajectories

The base of our approach is a main memory access structure based on a grid file. The grid partitions the $\mathbb{R} \times D$ domain into cells. Every cell $g[t_i, d_j]$ corresponds to a time period $i = [t_n, t_{n+k}]$ and an interval in the distance domain $j = [d_j, d_{j+h}]$.

In order to insert a KP-representation of a moving object into the grid, the intersected grid cells are determined (see Fig. 8a)). Every cell intersected by the KP representation of an object o stores its objectID o_{id} and itineraryID I_{id}. The KP-representation of a moving object is stored persistently for each object.



(a) Cells intersected by the KP-representations of object o_2

(b) Grid rows of correlation sections for itineraries o_i,k_i

Fig. 8. Grid-based representations of moving objects in the $\mathbb{R} \times D$ domain

Using a grid for representing trajectories leads to a crucial advantage. Due to the partitioning of the $\mathbb{R} \times D$ domain, the correlation sections of a relationship $d_{\otimes}(I_1, I_2)$, $d_{\otimes}(I_2, I_1)$ now correspond to rows of the grid (see Fig. 8b)). This, in turn, enables us to directly retrieve the relevant section in a KP representation by simply using the basic access functions of the grid. For example, for a given time t and the correlation sections $d_{\otimes}(I_1, I_2) \equiv n$, $d_{\otimes}(I_2, I_1) \equiv m$ the relevant sections of both itineraries correspond to the grid cells $g[t, n]$, $g[t, m]$. In contrast to the more complicated auxiliary function *clipping*() (see Sect. 5.3), the access function has a constant complexity. This makes the evaluation of relationships with correlations sections much more efficient.

In addition to the grid, we assume an *itinerary repository*. It stores the correlation sections between itineraries. Since the expected number of itineraries is relatively small, the correlation sections or rather their corresponding rows in the grid are pre-computed[4]. Thus, the repository stores a list of tuples $(I_1, [row_{start} - row_{end}], I_2, [row_{start} - row_{end}])$ for every relationship $\otimes$. Each tuple contains the correlation sections ($[row_{start} - row_{end}]$ and $[row_{start} - row_{end}]$) of Itineraries I_1 and I_2 for the relationship $\otimes$. In case of relationships which can also be hold among objects on the same itinerary (e.g. *overtake*), there exists a tuple $(I_1, [0 - row_{last}], I_1, [0 - row_{last}])$. It indicates that the relationship can be hold "everywhere" along the entire run of objects on the same itinerary.

6.2 Querying the Grid for Relationships

In general two kinds of queries for relationships can be distinguished: *point queries* and *range queries*.

Point Queries: A point query corresponds to the statement *"find all objects in the grid which hold the relationship $\otimes$ with a given object $O = (kpT, I_S)$"*. Typical of point queries is a given object, which has to be evaluated for a relationship with other objects. With respect to Sect. 1, the proposed queries are point queries, e.g. "*In which sections will the hazardous freight transport A pass passenger trains?*". In order to answer such queries four general steps are necessary:

1. The coordinates of the cells intersected by the KP-representation of the object $O = (kpT, I_S)$ are determined.
2. The relevant itineraries for the relationship $\otimes$ and given itinerary I_S are retrieved from the itinerary repository. The result is a list of tuples (I_S,$[row_{start_S} - row_{end_S}]$, I_N,$[row_{start_N} - row_{end_N}]$), which itinerary generally can hold the relationship $\otimes$ with I_S .
3. Basing on the list of tuples from the repository, the list of intersected grid cells of o_2 is scanned. Thereby, every coordinate of the list, i.e. it row value, is transformed into the corresponding correlation row of itinerary I_N, i.e.

[4] In case of a high number of itineraries pre-computing can be to expensive and "-on-demand-" computing may be more adequate. In [2] an approach has been proposed to efficiently find common sections of itineraries. By adopting this approach, also the determination of correlation sections is possible.

$row_{start_S} \equiv row_{start_N}$,...,$row_{end_S} \equiv row_{end_N}$. The transformed grid coordinates are used to access the grid. Every retrieved grid cell is scaned whether the list of stored objects contains objects on itinerary I_N. The result is a list of objectID.

4. The list of cells intersected by an object is a coarse approximation of its original run. Thus, the query result contains a certain amount of "false hits", which must be eliminated. Therefore, the persistent stored representations of the resulting objects are retrieved. Only if the right geometrical relationship of a relationship is hold with the given object O then this object is considered as a correct hit. In addition, comparing the original polylines, also enables us to answer queries in a qualitative way.

Range Queries: A range query corresponds to the statement "*retrieve all objects which hold the relationship* $\otimes$". In contrast to point queries, only the relationship and if at all a temporal interval is given, e.g. "*Which objects (will) overtake each other in the next 3 hours*" or "*Which objects from itineraries A and B (will) pass junction X in the next 2 days*".

In order to answer such queries, the same proceeding like for point queries can be used. However, in the second step the entire list of itineraries for a relationship $\otimes$ must be retrieved from the repository. If only some of the itineraries are relevant (see second example above) the list can be filtered in addition.

6.3 An Illustrating Example

Assume, we have the situation illustrated in Fig. 1 and want to answer the (point) query "*Will object* o_2 ***pass*** any *objects along its itinerary*?".

The relevant supporting points of the network sections shall have the euclidean coordinates $p_1 = (0, 7)$, $p_2 = (3, 3)$, $p_4 = (5, 3)$, $p_5 = (5, 1)$, $p_6 = (11, 1)$, $p_7 = (11, 3)$, $p_9 = (11, 7)$. There exist two itineraries o_i and k_i, with objects o_1, o_2, o_3,k_1, and k_2. Like in Fig. 8, we assume that each grid cell $g[t_i, d_j]$ represents an interval of 10 units in the time domain and an interval of 2 units in the distance domain. Because the temporal domain progresses continuously, a temporal "offset" is used to adjust temporal intervals accordingly. In our example, this offset is 5, i.e. the grid cell $g[1, 1]$ corresponds to the interval $[5, 15]$ in the temporal domain (and to the interval $[0, 2]$ in the distance domain).

As explained in Sect. 4 and 6.1, all correlation sections are pre-computed and stored as tuples in the itinerary repository. In our example, the relevant correlation section is $C_{pass}(o_i, k_i) = (p_4, p_7)$. To be stored in the itinerary respository, two things have to be done: It must be transformed into the $D \times \mathbb{R}$ domain and the corresponding grid coordinates (i.e., "grid rows") must be calculated. At first, the covered distance from the start point p_1 of the itinierary o_i to the start point p_4 of the correlation section is computed (cf. Def. 5). This is done as follows: Let $dist(p_1, p_2)$ be a function which calculates the euclidean distance between two points. If the distance from the start point p_1 of the itinerary to the start point p_4 of the correlation section is $dist(p_4, p_2) + dist(p_2, p_1) = 7$ and

to the end point p_7 is $dist(p_7, p_4) + dist(p_4, p_2) + dist(p_2, p_1) = 13$, then we obtain $d_{pass}(o_i, k_i) = (7, 13)$. Analogously, $d_{pass}(k_i, o_i) = (4, 10)$ for itinerary k_i are determined. At last, the correlation sections in the distance domain must be converted into the corresponding grid coordinates. With $D = 2$ being the length of the interval in the distance domain covered by an single grid cell, we obtain for $d_{pass}(o_i, k_i) = (7, 13)$ the grid rows $\lceil 7\ div\ (D = 2) \rceil = 4$ and $\lceil 13\ div\ (D = 2) \rceil = 7$. Analogously, the grid rows for $d_{pass}(k_i, o_i) = (4, 10)$ are determined. This leads to the entry $(o_i, [4-7], k_i, [3-5])$ in the itinerary repository. It means, that the correlations section $d_{pass}(o_i, k_i)$ regarding itinerary o_i affects the grid rows $[4-7]$ and $d_{pass}(k_i, o_i)$ regarding itinerary k_i the grid rows $[3-5]$ (cf. Fig. 8b)).

We assume, that objects k_1 and k_2 on itinerary k_i have the trajectories $T^{k_i} = (p_9, 30); (p_7, 40); (p_4, 50); (p_5, 55)$ and $T^{k_i} = ((p_9, 14); (p_7, 23);\ (p_4, 33); (p_5, 38))$, respectively. These trajectories have been transformed into their KP-representations (see later). Based on these KP-representations the affected grid cells are determined and the objectID and itineraryID are inserted. That is, e.g., grid cell $g[4, 6]$ receives the tuple (k_1, k_i), and grid cell $g[6, 5]$ the tuple (k_2, k_i) (cf. Fig. 8b)). In addition, the KP-representations of k_1 and k_2 are stored persistently in a database.

The object o_2, which is our object of interest, has the trajectory $T^{o_i} = ((p_1, 16);\ (p_2, 26); (p_4, 31); (p_7, 46); (p_6, 61))$ (i.e. o_2 starts at p_1 at time 16, reach p_2 at time 26, etc.). The first step to answer our query is to transform object o_2 into its KP-representation $o_2 = (o_i,\ kpT^{o_i} = (d_1 = 0, 16); (d_2 = 5, 26); (d_4 = 7, 31); (d_7 = 13, 46); (d_6 = 15, 61))$, where, e.g., $d_4 = dist(p_4, p_2) + dist(p_2, p_1)$ (see above). Based on the $KP-$representation the coordinates of the affected grid cells $Cells_{o_2}$ are determined (see highlighted cells in Fig. 8a)).

In the second step the relevant itineraries and correlation sections (grid rows) for the relationship *pass* and itinerary o_i are retrieved from the itinerary repository. In our example it is the single tuple $(o_i, [4-7], k_i, [3-5])$. In the third step, based on the list of grid coordinates $Cells_{o_2}$ the corresponding correlations sections or rather rows in the grid of itinerary k_i are determined. For the relationship *pass* solely the coordinates with the grid rows $[4-7]$ in $Cells_{o_2}$ are relevant, which are the coordinates $[3, 4], [3, 5], [4, 5], [4, 6], [4, 7], [5, 7]$ (see Fig. 8b)). For each of these grid coordinates, i.e. for their row value, the corresponding correlation row of itinerary k_i is determined. For example, the tuple $(o_i, [\mathbf{4}-7], k_i, [\mathbf{3}-5])$ indicates, that the grid coordinate $[3, \mathbf{4}]$ of o_2, i.e. its row value 4 has the correlation row 3 with respect to itinerary k_i. Thus the grid coordinate $[3, \mathbf{4}]$ is transformed into $[3, \mathbf{3}]$, and the coordinate $[5, \mathbf{7}]$ into the coordinate $[5, \mathbf{5}]$ (The first coordinate, i.e. the time coordinate remains the same). Using the transformed grid coordinates all affected grid cells are accessed and checked whether they contain objects on itinerary k_i. In our example, this leads to the result set (with duplicate elimination) $R = (k_1, k_2)$.

In order to eliminate "false hits" the persistently stored KP representation of object $k_1 = (k_i, kpT^{k_i} = (d_9 = 0, 14); (d_7 = 4, 23); (d_4 = 10, 33); (d_5 = 12, 38))$ and object $k_2 = (k_i, kpT^{k_i} = (d_9 = 0, 30); (d_7 = 4, 40); (d_4 = 10, 50); (d_5 =$

$12, 55)$) are retrieved. The relevant sections for the relationship *pass* are the polylines between the points $k_1' = (d_7 = 4, 23); (d_4 = 10, 33)$, $k_2' = (d_7 = 4, 40); (d_4 = 10, 50)$ and $o_2' = (d_4 = 7, 31); (d_7 = 13, 46)$. To check whether the relationship *pass* is "really" satisfied, the polylines of the relevant objects must *intersect*. To be able to perform this test, the polyline o_2' must be translated first (see Sect. 5.3). In our example, the translation vector is $\overrightarrow{v} = (0, -3)$, which leads in our example to the polyline $o_2'' = (d_4 = 4, 31); (d_7 = 10, 46)$. In addition, the polyline o_2'' must be mirrored to address the aspect of different movement directions on the same section (see Sect. 5.3). This leads to $o_2''' = (d_4 = 10, 31); (d_7 = 4, 46)$. Using the auxiliary function $intersect()$ leads to the intersection point $intersect(k_1, o_2''') = (9.52, 32.2)$ and $intersect(k_2, o_2''') = (5.44, 42.4)$ within the $D \times \mathbb{R}$ domain. Thus, both objects are correct hits and object o_2 *pass* objects k_1 and k_2. Transferred into the euclidean space this leads to the result, that o_2 *pass* object $k_1$on point $p_{pass}(o_2, k_1) = (5.48, 3)$ at time 32.2 which is shortly after point p_4. Object k_2 is passed by o_2 on point $p_{pass}(o_2, k_2) = (9.56, 3)$ at time 42.4 which is shortly before point p_7.

7 Conclusion

The proposed relationships enable the support of queries related to the dynamic behavior of moving objects on networks. Depending on the application, such relationships can hold either in the past, now or future.

For the efficient evaluation of relationships we used the distance-based representation of objects, i.e. their trajectories. This representation utilize the fact that objects on a network move along the same itinerary or identical sections. In consequence, commonly used spatial information of itineraries and thus of the network is kept only once and not for each object moving on it. This, in turn enables a compact representation of moving objects on this network.

In order to evaluate relationships also in case of large data sets, we transferred our results to a main memory access structure based on a grid file. The access structure considers the special properties of the distance-based domain and allows an adequate querying of relationships among moving objects on networks. We proposed algorithms for both point and range queries for relationships.

As part of our future work, we are going to identify and analyze alternative extensions to the grid in order to improve it selectivity. Other parts of our future work are the support of cyclic itineraries as well as the efficient support of dynamic changes of itineraries.

References

1. V.T. de Almeida and R.H. Güting. Indexing the trajectories of moving objects in networks. *Geoinformatica*, 9(1):33–60, March 2005.
2. Jae-Woo Chang and Jung-Ho Um. An efficient indexing scheme for moving objects' trajectories on road networks. In *Proc. of Advances in Web-Age Information Management (WAIM)*, pages 13–25, Hong Kong, China,, June 2006.

3. H. Don Chon, D. Agrawal, and A. El Abbadi. Storage and retrieval of moving objects. In *Proc. IEEE 2nd Int. Conf. on Mobile Data Management*, pages 173–184, Hong-Kong, China, 2001.
4. H. Don Chon, D. Agrawal, and A. El Abbadi. Using space-time grid for efficient management of moving objects. In *Proc. 2nd ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE)*, pages 59–65, Santa Barbara, USA, Mai 2001.
5. H. Don Chon, D. Agrawal, and A. El Abbadi. Query processing for moving objects with space-time grid storage model. In *Proc. Third International Conference on Mobile Data Management*, pages 121–128, Singapore,, Jan 2002.
6. H. Don Chon, D. Agrawal, and A. El Abbadi. Fates:finding a time dependent shortest path. In *Proc. 4th Int. Conf. on Mobile Data Management (MDM)*, pages 165–180, Melbourne, Australia, January 2003.
7. H. Don Chon, D. Agrawal, and A. El Abbadi. Range and knn query processing for moving objects in grid model. *Mobile Networks and Applications (MONET)*, 8(4):401–412, August 2003.
8. Max J. Egenhofer. Definition of line-line relations for geographic databases. *IEEE Data Engineering Bulletin*, 16(3):40–45, September 1993.
9. M. Erwig and M. Schneider. Spatio-temporal predicates. *IEEE Transactions on Knowledge and Data Engineering*, 14(4):881–901, July/August 2002.
10. E. Frentzos. Indexing objects moving on fixed networks. In *Proc. 8th Int. Symp. on Spatial and Temporal Databases (SSTD)*, pages 289–305, Santorini Island, Greece, July 2003.
11. R.H. Güting, V.T. de Almeida, and Z. Ding. Modeling and querying moving objects in networks. *The VLDB Journal*, 15(2):165–190, June 2006.
12. R.H. Güting and M. Schneider. *Moving Objects Databases.* Morgan Kaufmann Publishers, 2005.
13. C. S. Jensen, T. B. Pedersen, L. Speicys, and I. Timko. Data modeling for mobile services in the real world. In *Proc. 8th Int. Symp. on Spatial and Temporal Databases (SSTD)*, LNCS Vol. 2750, pages 1–9, Santorini, Greece, July 2003.
14. G. Kollios, D. Gunopolos, and V.J. Tsotras. On indexing mobile objects. In *Proc. 18th ACM SIGMOD-SIGACT-SIGART Symp. on Principles of Database System*, pages 261–272, Philadelphia, USA, May 1999.
15. G. Kollios, D. Gunopolos, V.J. Tsotras, A. Dellis, and M. Hadjieleftheriou. Indexing animated objects using spatiotemporal access methods. *IEEE Transactions on Knowledge and Data Engineering*, 13(5):758–777, September 2001.
16. G. Kollios, D. Papadopoulos, D. Gunopulos, and V. J. Tsotras. Indexing mobile objects using dual transformation. *The VLDB Journal The International Journal on Very Large Data Bases*, 14(2):238–256, April 2005.
17. Xiang Li and Hui Lin. Indexing networkconstrained trajectories for connectivity-based queries. *International Journal of Geographical Information Science*, 20(3):302–328, March 2006.
18. X. Meng and Z. Ding. Dsttmod: A discrete spatio-temporal trajectory based moving object database system. In *Proc. 14th Int. Conf. on Database and Expert Systems Applications (DEXA)*, LNCS 2736, pages 444–453, Prague, Czech Republic, September 2003.
19. D. Pfoser. Indexing the trajectories of moving objects. *IEEE Data Engineering Bulletin*, 25(2):3–9, June 2002.
20. D. Pfoser and C.S. Jensen. Indexing of network constrained moving objects. In *Proc. 11th ACM Int. Sym. on Advances in Geographic Information Systems (GIS)*, pages 25–32, New Orleans, USA, November 2003.

21. D. Pfoser, C.S. Jensen, and Y. Theodoridis. Novel approaches to the indexing of moving object trajectories. In *Proc. 26th Int. Conf. on Very Large Data Bases (VLDB)*, pages 395–406, Cairo, Egypt, September 2000.
22. K. Porkaew, I. Lazaridis, and S. Mehrotra. Querying mobile objects in spatio-temporal databases. In *Proc. Int. Symp. on Advances in Spatial and Temporal Databases (SSTD)*, pages 59–78, Redondo Beach, USA, July 2001.
23. S. Saltenis, C.S. Jensen, S.T. Leutenegger, and M. A. Lopez. Indexing the position of continuously moving objects. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 331–342, Dallas, USA, May 2000.
24. P. Scarponcini. Generalized model for linear referencing in transportation. *GeoInformatica*, 6(1):35–55, November 2002.
25. Baihua Zheng, Jianliang Xu, Wang-Chien Lee, and Lun Lee. Grid-partition index: a hybrid method for nearest-neighbor queries in wireless location-based services. *The VLDB Journal*, 15(1):21–39, January 2006.

Liste der bisher erschienenen Ulmer Informatik-Berichte

Einige davon sind per FTP von `ftp.informatik.uni-ulm.de` erhältlich
Die mit * markierten Berichte sind vergriffen

List of technical reports published by the University of Ulm

Some of them are available by FTP from `ftp.informatik.uni-ulm.de`
Reports marked with * are out of print

91-01 *Ker-I Ko, P. Orponen, U. Schöning, O. Watanabe*
Instance Complexity

91-02* *K. Gladitz, H. Fassbender, H. Vogler*
Compiler-Based Implementation of Syntax-Directed Functional Programming

91-03* *Alfons Geser*
Relative Termination

91-04* *J. Köbler, U. Schöning, J. Toran*
Graph Isomorphism is low for PP

91-05 *Johannes Köbler, Thomas Thierauf*
Complexity Restricted Advice Functions

91-06* *Uwe Schöning*
Recent Highlights in Structural Complexity Theory

91-07* *F. Green, J. Köbler, J. Toran*
The Power of Middle Bit

91-08* *V.Arvind, Y. Han, L. Hamachandra, J. Köbler, A. Lozano, M. Mundhenk, A. Ogiwara, U. Schöning, R. Silvestri, T. Thierauf*
Reductions for Sets of Low Information Content

92-01* *Vikraman Arvind, Johannes Köbler, Martin Mundhenk*
On Bounded Truth-Table and Conjunctive Reductions to Sparse and Tally Sets

92-02* *Thomas Noll, Heiko Vogler*
Top-down Parsing with Simulataneous Evaluation of Noncircular Attribute Grammars

92-03 *Fakultät für Informatik*
17. Workshop über Komplexitätstheorie, effiziente Algorithmen und Datenstrukturen

92-04* *V. Arvind, J. Köbler, M. Mundhenk*
Lowness and the Complexity of Sparse and Tally Descriptions

92-05* *Johannes Köbler*
Locating P/poly Optimally in the Extended Low Hierarchy

92-06* *Armin Kühnemann, Heiko Vogler*
Synthesized and inherited functions -a new computational model for syntax-directed semantics

92-07* *Heinz Fassbender, Heiko Vogler*
A Universal Unification Algorithm Based on Unification-Driven Leftmost Outermost Narrowing

92-08* *Uwe Schöning*
On Random Reductions from Sparse Sets to Tally Sets

92-09* *Hermann von Hasseln, Laura Martignon*
Consistency in Stochastic Network

92-10 *Michael Schmitt*
A Slightly Improved Upper Bound on the Size of Weights Sufficient to Represent Any Linearly Separable Boolean Function

92-11 *Johannes Köbler, Seinosuke Toda*
On the Power of Generalized MOD-Classes

92-12 *V. Arvind, J. Köbler, M. Mundhenk*
Reliable Reductions, High Sets and Low Sets

92-13 *Alfons Geser*
On a monotonic semantic path ordering

92-14* *Joost Engelfriet, Heiko Vogler*
The Translation Power of Top-Down Tree-To-Graph Transducers

93-01 *Alfred Lupper, Konrad Froitzheim*
AppleTalk Link Access Protocol basierend auf dem Abstract Personal Communications Manager

93-02 *M.H. Scholl, C. Laasch, C. Rich, H.-J. Schek, M. Tresch*
The COCOON Object Model

93-03 *Thomas Thierauf, Seinosuke Toda, Osamu Watanabe*
On Sets Bounded Truth-Table Reducible to P-selective Sets

93-04 *Jin-Yi Cai, Frederic Green, Thomas Thierauf*
On the Correlation of Symmetric Functions

93-05 *K.Kuhn, M.Reichert, M. Nathe, T. Beuter, C. Heinlein, P. Dadam*
A Conceptual Approach to an Open Hospital Information System

93-06 *Klaus Gaßner*
Rechnerunterstützung für die konzeptuelle Modellierung

93-07 *Ullrich Keßler, Peter Dadam*
Towards Customizable, Flexible Storage Structures for Complex Objects

94-01 *Michael Schmitt*
On the Complexity of Consistency Problems for Neurons with Binary Weights

94-02 *Armin Kühnemann, Heiko Vogler*
A Pumping Lemma for Output Languages of Attributed Tree Transducers

94-03 *Harry Buhrman, Jim Kadin, Thomas Thierauf*
On Functions Computable with Nonadaptive Queries to NP

94-04 *Heinz Faßbender, Heiko Vogler, Andrea Wedel*
Implementation of a Deterministic Partial E-Unification Algorithm for Macro Tree Transducers

94-05 *V. Arvind, J. Köbler, R. Schuler*
On Helping and Interactive Proof Systems

94-06 *Christian Kalus, Peter Dadam*
Incorporating record subtyping into a relational data model

94-07 *Markus Tresch, Marc H. Scholl*
A Classification of Multi-Database Languages

94-08 *Friedrich von Henke, Harald Rueß*
Arbeitstreffen Typtheorie: Zusammenfassung der Beiträge

94-09 *F.W. von Henke, A. Dold, H. Rueß, D. Schwier, M. Strecker*
Construction and Deduction Methods for the Formal Development of Software

94-10 *Axel Dold*
Formalisierung schematischer Algorithmen

94-11 *Johannes Köbler, Osamu Watanabe*
New Collapse Consequences of NP Having Small Circuits

94-12 *Rainer Schuler*
On Average Polynomial Time

94-13 *Rainer Schuler, Osamu Watanabe*
Towards Average-Case Complexity Analysis of NP Optimization Problems

94-14 *Wolfram Schulte, Ton Vullinghs*
Linking Reactive Software to the X-Window System

94-15 *Alfred Lupper*
Namensverwaltung und Adressierung in Distributed Shared Memory-Systemen

94-16 *Robert Regn*
Verteilte Unix-Betriebssysteme

94-17 *Helmuth Partsch*
Again on Recognition and Parsing of Context-Free Grammars:
Two Exercises in Transformational Programming

94-18 *Helmuth Partsch*
Transformational Development of Data-Parallel Algorithms: an Example

95-01 *Oleg Verbitsky*
On the Largest Common Subgraph Problem

95-02 *Uwe Schöning*
Complexity of Presburger Arithmetic with Fixed Quantifier Dimension

95-03 *Harry Buhrman,Thomas Thierauf*
The Complexity of Generating and Checking Proofs of Membership

95-04 *Rainer Schuler, Tomoyuki Yamakami*
Structural Average Case Complexity

95-05 *Klaus Achatz, Wolfram Schulte*
Architecture Indepentent Massive Parallelization of Divide-And-Conquer Algorithms

95-06 *Christoph Karg, Rainer Schuler*
Structure in Average Case Complexity

95-07 *P. Dadam, K. Kuhn, M. Reichert, T. Beuter, M. Nathe*
ADEPT: Ein integrierender Ansatz zur Entwicklung flexibler, zuverlässiger kooperierender Assistenzsysteme in klinischen Anwendungsumgebungen

95-08 *Jürgen Kehrer, Peter Schulthess*
Aufbereitung von gescannten Röntgenbildern zur filmlosen Diagnostik

95-09 *Hans-Jörg Burtschick, Wolfgang Lindner*
On Sets Turing Reducible to P-Selective Sets

95-10 *Boris Hartmann*
Berücksichtigung lokaler Randbedingung bei globaler Zieloptimierung mit neuronalen Netzen am Beispiel Truck Backer-Upper

95-12 *Klaus Achatz, Wolfram Schulte*
Massive Parallelization of Divide-and-Conquer Algorithms over Powerlists

95-13 *Andrea Mößle, Heiko Vogler*
Efficient Call-by-value Evaluation Strategy of Primitive Recursive Program Schemes

95-14 *Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß*
A Generic Specification for Verifying Peephole Optimizations

96-01 *Ercüment Canver, Jan-Tecker Gayen, Adam Moik*
Formale Entwicklung der Steuerungssoftware für eine elektrisch ortsbediente Weiche mit VSE

96-02 *Bernhard Nebel*
Solving Hard Qualitative Temporal Reasoning Problems: Evaluating the Efficiency of Using the ORD-Horn Class

96-03 *Ton Vullinghs, Wolfram Schulte, Thilo Schwinn*
An Introduction to TkGofer

96-04 *Thomas Beuter, Peter Dadam*
Anwendungsspezifische Anforderungen an Workflow-Mangement-Systeme am Beispiel der Domäne Concurrent-Engineering

96-05 *Gerhard Schellhorn, Wolfgang Ahrendt*
Verification of a Prolog Compiler - First Steps with KIV

96-06 *Manindra Agrawal, Thomas Thierauf*
Satisfiability Problems

96-07 *Vikraman Arvind, Jacobo Torán*
A nonadaptive NC Checker for Permutation Group Intersection

96-08 *David Cyrluk, Oliver Möller, Harald Rueß*
An Efficient Decision Procedure for a Theory of Fix-Sized Bitvectors with Composition and Extraction

96-09 *Bernd Biechele, Dietmar Ernst, Frank Houdek, Joachim Schmid, Wolfram Schulte*
Erfahrungen bei der Modellierung eingebetteter Systeme mit verschiedenen SA/RT–Ansätzen

96-10 *Falk Bartels, Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß*
Formalizing Fixed-Point Theory in PVS

96-11 *Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß*
Mechanized Semantics of Simple Imperative Programming Constructs

96-12 *Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß*
Generic Compilation Schemes for Simple Programming Constructs

96-13 *Klaus Achatz, Helmuth Partsch*
From Descriptive Specifications to Operational ones: A Powerful Transformation Rule, its Applications and Variants

97-01 *Jochen Messner*
Pattern Matching in Trace Monoids

97-02 *Wolfgang Lindner, Rainer Schuler*
A Small Span Theorem within P

97-03 *Thomas Bauer, Peter Dadam*
A Distributed Execution Environment for Large-Scale Workflow Management Systems with Subnets and Server Migration

97-04 *Christian Heinlein, Peter Dadam*
Interaction Expressions - A Powerful Formalism for Describing Inter-Workflow Dependencies

97-05 *Vikraman Arvind, Johannes Köbler*
On Pseudorandomness and Resource-Bounded Measure

97-06 *Gerhard Partsch*
Punkt-zu-Punkt- und Mehrpunkt-basierende LAN-Integrationsstrategien für den digitalen Mobilfunkstandard DECT

97-07 *Manfred Reichert, Peter Dadam*
$ADEPT_{flex}$ - Supporting Dynamic Changes of Workflows Without Loosing Control

97-08 *Hans Braxmeier, Dietmar Ernst, Andrea Mößle, Heiko Vogler*
The Project NoName - A functional programming language with its development environment

97-09 *Christian Heinlein*
Grundlagen von Interaktionsausdrücken

97-10 *Christian Heinlein*
Graphische Repräsentation von Interaktionsausdrücken

97-11 *Christian Heinlein*
Sprachtheoretische Semantik von Interaktionsausdrücken

97-12 *Gerhard Schellhorn, Wolfgang Reif*
Proving Properties of Finite Enumerations: A Problem Set for Automated Theorem Provers

97-13 *Dietmar Ernst, Frank Houdek, Wolfram Schulte, Thilo Schwinn*
Experimenteller Vergleich statischer und dynamischer Softwareprüfung für eingebettete Systeme

97-14 *Wolfgang Reif, Gerhard Schellhorn*
Theorem Proving in Large Theories

97-15 *Thomas Wennekers*
Asymptotik rekurrenter neuronaler Netze mit zufälligen Kopplungen

97-16 *Peter Dadam, Klaus Kuhn, Manfred Reichert*
Clinical Workflows - The Killer Application for Process-oriented Information Systems?

97-17 *Mohammad Ali Livani, Jörg Kaiser*
EDF Consensus on CAN Bus Access in Dynamic Real-Time Applications

97-18 *Johannes Köbler,Rainer Schuler*
Using Efficient Average-Case Algorithms to Collapse Worst-Case Complexity Classes

98-01 *Daniela Damm, Lutz Claes, Friedrich W. von Henke, Alexander Seitz, Adelinde Uhrmacher, Steffen Wolf*
Ein fallbasiertes System für die Interpretation von Literatur zur Knochenheilung

98-02 *Thomas Bauer, Peter Dadam*
Architekturen für skalierbare Workflow-Management-Systeme - Klassifikation und Analyse

98-03 *Marko Luther, Martin Strecker*
A guided tour through *Typelab*

98-04 *Heiko Neumann, Luiz Pessoa*
Visual Filling-in and Surface Property Reconstruction

98-05 *Ercüment Canver*
Formal Verification of a Coordinated Atomic Action Based Design

98-06 *Andreas Küchler*
On the Correspondence between Neural Folding Architectures and Tree Automata

98-07 *Heiko Neumann, Thorsten Hansen, Luiz Pessoa*
Interaction of ON and OFF Pathways for Visual Contrast Measurement

98-08 *Thomas Wennekers*
Synfire Graphs: From Spike Patterns to Automata of Spiking Neurons

98-09 *Thomas Bauer, Peter Dadam*
Variable Migration von Workflows in *ADEPT*

98-10 *Heiko Neumann, Wolfgang Sepp*
Recurrent V1 – V2 Interaction in Early Visual Boundary Processing

98-11 *Frank Houdek, Dietmar Ernst, Thilo Schwinn*
Prüfen von C–Code und Statmate/Matlab–Spezifikationen: Ein Experiment

98-12 *Gerhard Schellhorn*
Proving Properties of Directed Graphs: A Problem Set for Automated Theorem Provers

98-13 *Gerhard Schellhorn, Wolfgang Reif*
Theorems from Compiler Verification: A Problem Set for Automated Theorem Provers

98-14 *Mohammad Ali Livani*
SHARE: A Transparent Mechanism for Reliable Broadcast Delivery in CAN

98-15 *Mohammad Ali Livani, Jörg Kaiser*
Predictable Atomic Multicast in the Controller Area Network (CAN)

99-01 *Susanne Boll, Wolfgang Klas, Utz Westermann*
A Comparison of Multimedia Document Models Concerning Advanced Requirements

99-02 *Thomas Bauer, Peter Dadam*
Verteilungsmodelle für Workflow-Management-Systeme - Klassifikation und Simulation

99-03 *Uwe Schöning*
On the Complexity of Constraint Satisfaction

99-04 *Ercument Canver*
Model-Checking zur Analyse von Message Sequence Charts über Statecharts

99-05 *Johannes Köbler, Wolfgang Lindner, Rainer Schuler*
Derandomizing RP if Boolean Circuits are not Learnable

99-06 *Utz Westermann, Wolfgang Klas*
Architecture of a DataBlade Module for the Integrated Management of Multimedia Assets

99-07 *Peter Dadam, Manfred Reichert*
Enterprise-wide and Cross-enterprise Workflow Management: Concepts, Systems, Applications. Paderborn, Germany, October 6, 1999, GI–Workshop Proceedings, Informatik ’99

99-08 *Vikraman Arvind, Johannes Köbler*
Graph Isomorphism is Low for ZPP^{NP} and other Lowness results

99-09 *Thomas Bauer, Peter Dadam*
Efficient Distributed Workflow Management Based on Variable Server Assignments

2000-02 *Thomas Bauer, Peter Dadam*
Variable Serverzuordnungen und komplexe Bearbeiterzuordnungen im Workflow-Management-System ADEPT

2000-03 *Gregory Baratoff, Christian Toepfer, Heiko Neumann*
Combined space-variant maps for optical flow based navigation

2000-04 *Wolfgang Gehring*
Ein Rahmenwerk zur Einführung von Leistungspunktsystemen

2000-05 *Susanne Boll, Christian Heinlein, Wolfgang Klas, Jochen Wandel*
Intelligent Prefetching and Buffering for Interactive Streaming of MPEG Videos

2000-06 *Wolfgang Reif, Gerhard Schellhorn, Andreas Thums*
Fehlersuche in Formalen Spezifikationen

2000-07 *Gerhard Schellhorn, Wolfgang Reif (eds.)*
FM-Tools 2000: The 4th Workshop on Tools for System Design and Verification

2000-08 *Thomas Bauer, Manfred Reichert, Peter Dadam*
Effiziente Durchführung von Prozessmigrationen in verteilten Workflow-Management-Systemen

2000-09 *Thomas Bauer, Peter Dadam*
Vermeidung von Überlastsituationen durch Replikation von Workflow-Servern in ADEPT

2000-10 *Thomas Bauer, Manfred Reichert, Peter Dadam*
Adaptives und verteiltes Workflow-Management

2000-11 *Christian Heinlein*
Workflow and Process Synchronization with Interaction Expressions and Graphs

2001-01 *Hubert Hug, Rainer Schuler*
DNA-based parallel computation of simple arithmetic

2001-02 *Friedhelm Schwenker, Hans A. Kestler, Günther Palm*
3-D Visual Object Classification with Hierarchical Radial Basis Function Networks

2001-03 *Hans A. Kestler, Friedhelm Schwenker, Günther Palm*
RBF network classification of ECGs as a potential marker for sudden cardiac death

2001-04 *Christian Dietrich, Friedhelm Schwenker, Klaus Riede, Günther Palm*
Classification of Bioacoustic Time Series Utilizing Pulse Detection, Time and Frequency Features and Data Fusion

2002-01 *Stefanie Rinderle, Manfred Reichert, Peter Dadam*
Effiziente Verträglichkeitsprüfung und automatische Migration von Workflow-Instanzen bei der Evolution von Workflow-Schemata

2002-02 *Walter Guttmann*
Deriving an Applicative Heapsort Algorithm

2002-03 *Axel Dold, Friedrich W. von Henke, Vincent Vialard, Wolfgang Goerigk*
A Mechanically Verified Compiling Specification for a Realistic Compiler

2003-01 *Manfred Reichert, Stefanie Rinderle, Peter Dadam*
A Formal Framework for Workflow Type and Instance Changes Under Correctness Checks

2003-02 *Stefanie Rinderle, Manfred Reichert, Peter Dadam*
Supporting Workflow Schema Evolution By Efficient Compliance Checks

2003-03 *Christian Heinlein*
Safely Extending Procedure Types to Allow Nested Procedures as Values

2003-04 *Stefanie Rinderle, Manfred Reichert, Peter Dadam*
On Dealing With Semantically Conflicting Business Process Changes.

2003-05 *Christian Heinlein*
Dynamic Class Methods in Java

2003-06 *Christian Heinlein*
Vertical, Horizontal, and Behavioural Extensibility of Software Systems

2003-07 *Christian Heinlein*
Safely Extending Procedure Types to Allow Nested Procedures as Values
(Corrected Version)

2003-08 *Changling Liu, Jörg Kaiser*
Survey of Mobile Ad Hoc Network Routing Protocols)

2004-01 *Thom Frühwirth, Marc Meister (eds.)*
First Workshop on Constraint Handling Rules

2004-02 *Christian Heinlein*
Concept and Implementation of C+++, an Extension of C++ to Support User-Defined Operator Symbols and Control Structures

2004-03 *Susanne Biundo, Thom Frühwirth, Günther Palm(eds.)*
Poster Proceedings of the 27th Annual German Conference on Artificial Intelligence

2005-01 *Armin Wolf, Thom Frühwirth, Marc Meister (eds.)*
19th Workshop on (Constraint) Logic Programming

2005-02 *Wolfgang Lindner (Hg.), Universität Ulm , Christopher Wolf (Hg.) KU Leuven*
2. Krypto-Tag – Workshop über Kryptographie, Universität Ulm

2005-03 *Walter Guttmann, Markus Maucher*
Constrained Ordering

2006-01 *Stefan Sarstedt*
Model-Driven Development with ACTIVECHARTS, Tutorial

2006-02 *Alexander Raschke, Ramin Tavakoli Kolagari*
Ein experimenteller Vergleich zwischen einer plan-getriebenen und einer leichtgewichtigen Entwicklungsmethode zur Spezifikation von eingebetteten Systemen

2006-03 *Jens Kohlmeyer, Alexander Raschke, Ramin Tavakoli Kolagari*
Eine qualitative Untersuchung zur Produktlinien-Integration über Organisationsgrenzen hinweg

2006-04 *Thorsten Liebig*
Reasoning with OWL - System Support and Insights –

2008-01 *H.A. Kestler, J. Messner, A. Müller, R. Schuler*
On the complexity of intersecting multiple circles for graphical display

2008-02 *Manfred Reichert, Peter Dadam, Martin Jurisch,l Ulrich Kreher, Kevin Göser, Markus Lauer*
Architectural Design of Flexible Process Management Technology

2008-03 *Frank Raiser*
Semi-Automatic Generation of CHR Solvers from Global Constraint Automata

2008-04 *Ramin Tavakoli Kolagari, Alexander Raschke, Matthias Schneiderhan, Ian Alexander*
Entscheidungsdokumentation bei der Entwicklung innovativer Systeme für produktlinien-basierte Entwicklungsprozesse

2008-05 *Markus Kalb, Claudia Dittrich, Peter Dadam*
Support of Relationships Among Moving Objects on Networks

Ulmer Informatik-Berichte
ISSN 0939-5091

Herausgeber:
Universität Ulm
Fakultät für Ingenieurwissenschaften und Informatik
89069 Ulm