A Generic Data Model for Moving Objects

Jianqiu Xu, Ralf Hartmut Güting

Database Systems for New Applications, Mathematics and Computer Science
FernUniversität Hagen, Germany
{jianqiu.xu,rhg}@fernuni-hagen.de

May 24, 2012

Abstract

Moving objects databases should be able to manage trips that pass through several real world environments, e.g., road network, indoor. However, the current data models only deal with the movement in one situation and cannot represent comprehensive trips for humans who can move inside a building, walk on the pavement, drive on the road, take the public vehicles (bus or train), etc. As a result, existing queries are solely limited to one environment. In this paper, we design a data model that is able to represent moving objects in multiple environments in order to support novel queries on trips in different surroundings and various transportation modes (e.g., *Car*, *Walk*, *Bus*). A generic and precise location representation is proposed that can apply in all environments. The idea is to let the space for moving objects be covered by a set of so-called infrastructures each of which corresponds to an environment and defines the available places for moving objects. Then, the location is represented by referencing to the infrastructure. We formulate the concept of space and infrastructure and propose the methodology to represent moving objects in different environments with the integration of precise transportation modes. Due to different infrastructure characteristics, a set of novel data types is defined to represent infrastructure components. To efficiently support new queries, we design a group of operators to access the data. We present how such a data model is implemented in a database system and report the experimental results.

The new model is designed with attention to the data models of previous work for free space and road networks to have a consistent type system and framework of operators. In this way, a powerful set of generic query operations is available for querying, together with those dealing with infrastructures and transportation modes. We demonstrate these capabilities by formulating a set of sophisticated queries across all infrastructures.

1 Introduction

Moving objects databases have been extensively studied in the last decade due to their wide applications, e.g., location-based services [22, 58, 24], transportation and road networks [34, 43, 9, 33], nearest neighbor queries [47, 23, 32, 19], and trajectory searching [11, 28, 12]. Basically, a moving objects database manages spatial objects continuously changing locations over time. To fully understand the mobility of a traveler, recently researchers started to explore the area of moving objects with different transportation modes [60, 59, 39, 45]. People try to discover movement segments with determined motion modes from raw GPS data such as walking, driving, taking a bus as these pieces of information denote important characteristics of a mobile user's context and can enrich the mobility with informative and context knowledge. To completely identify the movement for a person, two factors are needed: where and how. In a transportation system, providing a trip with different motion modes and the constraint on a mode (e.g., less than two bus transfers) is substantially meaningful for a traveler [9]. In respect of all the above work, taking into account transportation modes becomes increasingly important for moving objects, and new challenges are imposed for a database system to efficiently manage trips passing through several environments.

1.1 Motivation

To motivate the scope of this paper, consider the following two movement scenarios of *Bobby*:

M_1: *walks from his house to the parking lot, and then drives the car along the road and highway to his office building, finally walks from the underground garage to his office room.*

M_2: *walks from the house to a bus stop, and then takes a bus to the train station, moves from one city to another by train, finally walks from the train station to his office room.*

The two trips can be described by a sequence of transportation modes where (1) M_1: *Walk* $\rightarrow$ *Car* $\rightarrow$ *Indoor*; (2) M_2: *Walk* $\rightarrow$ *Bus* $\rightarrow$ *Train* $\rightarrow$ *Walk* $\rightarrow$ *Indoor*. Each trip passes several environments and each environment owns its possible transportation modes. Such comprehensive movements need to be efficiently and effectively managed by a database system. Existing data models [42, 20, 14, 22, 44, 18] for moving objects only address the issue in a specific environment and cannot represent the comprehensive trips above. We classify the current state-of-the-art into three categories:

1. free space [42, 53, 14, 20, 16];
2. road (spatial) network [50, 22, 44, 18];
3. indoor [26, 25].

Free space is an environment with no movement constraint. But in practice, objects usually move on a pre-defined set of trajectories as specified by the underlying network (road, highway, etc). The first two are for outdoor. In daily life, people also spend a large amount of time in indoor space such as office buildings, shopping centers, etc. According to the Nokia report [2], the percentage of time for people staying inside buildings (GPS activates only outdoor) is up to 90%. The models above proposed different techniques for location representation and data manipulation. Each model can only manage the data limited to one environment. There is no management for different environments in a global system and the relationship between them is not considered, for example, the places where people can switch the transportation mode such as bus stops and building entrances. A complete trip with several transportation modes like M_1 (M_2) is not managed by previous techniques. For different applications, the movement is split into several parts each of which fits into one environment.

Due to the limitation of data model, interesting queries regarding transportation modes and environments cannot be answered. Consider the following two examples.

Q_1:"*where is Bobby at 8am, e.g., at home, in the street or in the office room?*";

Q_2: "*how long does Bobby walk during his trip?*"

If a complete trip is not managed, Q_1 and Q_2 cannot be supported in one system. To answer Q_1, one first has to identify the environment that *Bobby* belongs to at the given time instant. For Q_2, the walk movement might have several parts located in different places (outdoor and indoor). One first has to find all walking parts in the whole trip. Most existing methods [42, 53, 14, 20] use a pair (x, y) to identify a location; obviously, (x, y) cannot apply for indoor as it is a 3D environment. Of course, it is possible to extend the pair to a triple (x, y, z) to represent the location. But the method only focuses on geometric properties (e.g., coordinates). The raw location data (x, y, z) means nothing else than three real numbers so that it is unable to recognize the part for walking (or driving). Usually, a person's trip has several transportation modes, e.g., *Bus*, *Walk*, and the trip should be represented in such a way that (1) a complete movement is managed; (2) a precise location in each environment is defined; and (3) one can efficiently retrieve a sub trip according to the transportation mode and distinguish trips with different modes.

Besides free space, road network and indoor, there are still two environments that receive little attention: Public Transportation Network (PTN) and Region-based Outdoor (RBO). PTN seems to be similar as road network because both have pre-defined paths for moving objects. But there are still significant differences: (1) the movement in PTN is not only limited to bus routes but also depends on time schedules; (2) in a road network one can move any distance along the road and change the direction at any junction if allowed, but bus travelers can only start and end their movement at bus stops, sole places for transfer. RBO represents the area for people walking outside such as pavement and footpath. The overall area in a city for outdoor walking can be considered as a relatively large polygon with many obstacles inside. The obstacles denote places such as building blocks and junctions areas. People can move freely inside the polygon but not directly pass through obstacles.

To manage a comprehensive movement in a database system, a data model that is able to represent moving objects in all available environments is needed. Although environments have different characteristics such as constrained or free, 2D or 3D, and movement depending on other vehicles, the location and moving objects representation should be robust and general in order to apply for all cases.

1.2 Data Models for Moving Objects

Basically, there are two contrasting approaches to modeling geometric location data [41]: *field-based* and *object-based* models. The first one considers the space as being continuous and empty, and there are discrete entities (objects) moving inside the space having their own properties, i.e., coordinates. References [42, 53, 14, 20] belong to this category. The other method deals with the world as a surface littered with recognizable and geographic objects that are associated with spatial attributes, and every location in space is represented by mapping to those objects. Papers [40, 51, 10, 35, 18] fall into this category. In this paper, we follow the method of the *object-based* approach but model all environments rather than one. We let the geographic space be covered by a set of so-called infrastructures, denoting all real world environments. Five infrastructures are considered in total: (1) Public Transportation Network, (2) Region-based Outdoor, (3) Free Space, (4) Road Network, and (5) Indoor. Each infrastructure consists of a set of components called infrastructure objects (IFOBs) that represent available places for moving objects. For example, in a road network, IFOBs are roads and streets. Pavements and footpaths represented by polygons constitute RBO. The location of moving objects is represented by referencing to these IFOBs.

The continuously changing location data is abstractly described by a function projecting from time to a location. However, at present the representation is different depending on the environment feature. In free space, a pair (x, y) [42, 53, 14, 20] is used to identify a location. A road network position is denoted by (rid, pos) [18] where rid ($\in D_{\underline{int}}$) illustrates a road identifier and pos ($\in D_{\underline{real}}$) records the position on the road. Compared with outdoor, an indoor model should consider both horizontal and vertical positions to uniquely identify a location.

To manage moving objects in different environments, the location representation should be general and consistent in all cases rather than limited to one. At the same time, the unique feature of each individual situation should not be lost. There are two challenges (1) IFOBs (e.g., roads, polygons) have diverse characteristics and are represented by different data types in the database system; (2) the location is related to these heterogeneous IFOBs. Additionally, although there is no IFOB in free space, the location model must maintain its feature in order to seamlessly integrate such an environment, as the goal is to represent moving objects in all cases. We design the novel data model in such a way that on one hand a generic method is proposed to represent the precise location in all infrastructures. On the other hand, the location representation of previous work in free space and road network [1] can be integrated so that former techniques are directly supported without too much effort. As a result, the generic location representation regresses to specific representation in free space and road network and previous models are integrated as two infrastructures into the new model. One does not have to design new data types and operators for environments that have already been well established. The new model is not simply a wrapper because (i) PTN and RBO are not treated by previous techniques but modeled by our method; (ii) indoor location is precisely represented.

To manage generic moving objects in a database system, the data model should define (1) a robust method that can represent the continuously changing location in all available environments; (2) a set of data types representing IFOBs and moving objects; (3) a group of operators for efficiently accessing and manipulating the data. We have explained the reference idea for (1) above. Since environments have different characteristics, IFOBs in each case are represented by different data types in the database system. It is straightforward that a line represents a road and a polygon denotes the area for a pavement. But for PTN and Indoor, new data types are needed as objects have special features that cannot be represented by existing data types. To efficiently support query processing, operators have to be defined in the system for users to access the data and formulate queries. As a result, Sec. 6 and 7 demonstrate a comprehensive set of example queries ranging over different infrastructures and transportation

[1] indoor location is not precisely represented by existing models

modes.

1.3 Contribution

The main contribution of this paper is the design of a generic data model for managing moving objects in various environments. It includes the following specific contributions:

- We formulate the concept of space and infrastructure and define their components. A general definition for heterogeneous IFOBs is given. We propose a method to represent the location of moving objects in all defined environments. Moreover, a framework for moving objects representation is designed.
- We model available places for each infrastructure and give the data type representing its components, i.e., IFOBs. Applying the proposed framework, we present how the location is represented in each infrastructure. For the indoor environment, we define a graph model for indoor navigation that supports optimal route searching with respect to different costs (e.g., distance, time).
- A type system is defined for the generic model, and a relational interface is provided to exchange information. A set of operators is designed on the proposed data types. The syntax and semantics of operators are also defined. Type system and operations are carefully designed to (1) obtain a powerful query language; and (2) integrate the well-established models of free space and road network.
- A set of example queries on all available environments and various transportation modes is formulated, demonstrating the expressiveness of the resulting querying framework. We present how such a data model is implemented in a database system. Query optimization techniques are also introduced. We report the experimental results of running proposed queries in a database system.

The rest of the paper is organized as follows: Section 2 reviews the related work. The framework of the data model is presented in Section 3. Section 4 describes data types for infrastructure components and the location representation in each case. Section 5 defines the type system and provides a relational interface. The proposed operators are presented in Section 6. A comprehensive set of queries is formulated in Section 7. In Section 8 we describe the implementation of the data model. Section 9 reports the experimental results. Finally, Section 10 concludes the paper.

2 Related Work

2.1 Modeling Moving Objects

In the database literature, there has been a large body of recent works [53, 20, 46, 52, 22, 44, 10, 13, 35, 18, 29, 37, 38, 27, 25] on modeling moving objects. But all of them deal with the movement in one environment and do not investigate transportation modes. The closest to our work is [10, 35, 18] where the models consider the underlying environment for location representation. In [10], a semantic model is proposed for representing trajectories based on background geographic information. An algorithm is developed to map the positions of vehicles into a road network. Thus, the spatial aspect of trajectories is modeled in terms of a network, which consists of edges and nodes. However, the method is restricted to a specific application domain. To answer mobility pattern queries [35], a model is designed that relies on a discrete view of the underlying space for moving objects. The authors partition the space into a set of zones each of which is uniquely identified by a label. Afterwards, the location is represented by mapping it into zones, and a trajectory is defined as a sequence of labels. A so-called *route-oriented* model is presented in [18] for moving objects in networks. The method represents a road network by routes and junctions, and trajectories are integrated with the road network. A line is used to describe the geometrical property of a road. Then, a network location is represented by a road id and the position on the road. However, the aforementioned models have the following drawbacks. First, the methods can only represent the location in one environment, making the model not general. A complete trip passing several environments cannot be managed in a database

system. Second, the location in [35] is not precisely represented, only identified by a symbol pointing to a zone. The model cannot answer a precise location query. Furthermore, moving objects are represented in a discrete way (a sequence of timestamps) instead of continuous. The method is limited to one application. Third, transportation modes are not handled.

GTFS (General Transit Feed Specification) [1] defines a common format for public transportation schedules and associated geographic information, so that people can use the GTFS specification to provide schedules and geographic information to Google Maps. The specification contains some files (required and optional), and each file with a defined format records a list of items each of which stores a field name and value. For example, a stop file includes the following required field names: *stop_id*, *stop_name*, *stop_lat*, *stop_lon*. However, the representation for moving buses is not included. Locations of a bus traveler are a set of items recording bus stops as well as arrival and departure time at each stop, while positions between two *adjacent* stops are not determined. In fact, the locations of a bus traveler depend on the bus and do not have to be additionally represented.

GPS is the dominant positioning technology for outdoor settings. For indoor environment, new techniques are required. A graph model is proposed in [25] for indoor tracking moving objects using the technology RFID. They assume RFID readers are embedded in the indoor space in some known positions and the indoor space is partitioned into cells corresponding to vertices in the graph. An edge in the graph indicates the movement between cells which is detected by RFID readers. A raw trajectory is a sequence of RFID tags. A method is developed to construct and refine the trajectory. The goal is to improve the indoor tracking accuracy. The result is different from the intention of this paper, modeling moving objects. *Jensen et al.* [26] present an index structure for moving objects in a symbolic indoor space. The trajectory model is composed of records in the format $(oid, symbolicID, t)$ where oid is the moving object identifier, $symbolicID$ is the identifier for a specific indoor space region and t indicates time. Nevertheless, the authors there do not give the data type representing a space region and define the location in an imprecise way. The room where the object is located can be known, but the precise location inside cannot be determined.

A multidimensional data model is proposed in [49] to provide a foundation for capturing and querying complex transportation infrastructures. The model captures important transportation infrastructure concepts such as roads, road part, lanes and the relationships among them. Each individual lane is captured separately due to different road characteristics, and the relationship containment is captured among segments in different levels. Different contents are attached to some specific points and road sections, e.g., traffic accidents, gas stations, speed limits. Based on the model in [24], the representation dimension is extended by introducing three new relations on dimension values to capture direction, traffic exchange and lane change relationships between road segments. Some properties of the above three relations between segments are defined such as transitivity and propagation of direction. Each representation of the transportation infrastructure is modeled as a separate dimension hierarchy. Multiple representations are also proposed. Two categories of queries are considered: transportation infrastructure queries and dynamic queries. However, only the road network is modeled without considering the other infrastructures such as public transportation system. The indoor environment is not handled. Besides, the representation of moving objects in both outdoor and indoor spaces is not defined.

2.2 Transportation Modes

A data model presented in [9] gives the framework of a transportation system. The model is able to provide a trip consisting of several transportation modes, e.g., *Bus*, *Walk*, *Train*. Moving objects databases and graph-based databases are integrated to facilitate trip planning in urban transportation networks. The authors deal with returning a shortest path (SP) with multiple transportation modes to connect the origin and the destination, where SP can have more constraints and choices, e.g., different motion modes, the number of transfers. A graph model is defined where each vertex corresponds to a place in a transportation network. The place has a name and a geometric representation, e.g., point or region. Each edge is associated with a transportation mode. Edges with different modes can be incident on the same vertices indicating that a transfer between different modes can happen. A trip is defined as a sequence of *legs* each of which represents a path with one transportation mode. However, the multimodal model aims to provide trip plannings with various constraints rather than model moving objects

in various environments. The proposed *leg* and multimodal *trip* are not represented by defining data types in a database system, but described conceptually and abstractly. Besides, indoor environment is not included. We focus on representing moving objects in all real world environments and managing multimodal trips in a database system so that users can request queries in a system context. In our model, transportation modes are seamlessly integrated into moving objects. An interesting query is proposed in [8] that computes isochrones in multi-model and schedule-based transport networks. The goal is to find the set of points on a road network, from which a specific point of interest can be reached within a given time span. Only the transportation modes *Walk* and *Bus* are considered.

In Microsoft's project GeoLife, the work [60, 59] aims to discover and infer transportation modes from raw GPS trajectory data. By mining multiple users' location histories, one can discover the most interesting locations, classical travel sequences and travel experts in a given geospatial region, hence enabling a generic travel recommendation. The procedure comprises three phases. First, a GPS trajectory is decomposed into several segments of different transportation modes, while maintaining a segment of one mode as long as possible. A set of features being independent of the velocity is identified. Second, the determined features are fed into a classification model to output the probability of each segment with different transportation modes. Third, a graph-based postprocessing algorithm is developed to further improve the inference performance. Some researchers utilize mobile phones to detect transportation modes when outside. [39] creates a classification system that uses a mobile phone with a built-in GPS receiver and an accelerometer to determine the transportation mode of an individual when outside. *Stenneth et al.* [45] proposed an approach to inferring a user's mode based on the GPS sensor on the mobile device as well as the knowledge of the underlying transportation network, e.g., bus stop locations, railway lines. The above work is different from ours. We concentrate on representing moving objects in various environments with transportation modes instead of inferring the modes. In addition, they only take into consideration outdoor movement because the inferring method is based on GPS data where a GPS receiver will lose signal indoors.

3 Generic Data Model

3.1 Preliminaries

We give the carrier set of basic types that we use for the definitions in the following sections [2].

Definition 3.1 *Base Types*
$D_{\underline{int}} = \mathbb{Z} \cup \{\bot\}$, $D_{\underline{real}} = \mathbb{R} \cup \{\bot\}$, $D_{\underline{bool}}$ = {*FALSE, TRUE*} $\cup \{\bot\}$

Each domain is based on the usual interpretation with an extension by the undefined value, denoted by $\bot$. We give three data types representing time: *time instant*, *time interval* and *time range*, defined as follows:

Definition 3.2 *Time Types*
time instant: $D_{\underline{instant}} = \mathbb{R} \cup \{\bot\}$
time interval: $D_{\underline{interval}}$ = $\{(s, e, lc, rc) | s, e \in D_{\underline{instant}}, lc, rc \in D_{\underline{bool}}, s \leq e,$
$(s = e) \Rightarrow (lc = rc = true)\}$
time range: $D_{\underline{periods}} = \{V \subseteq D_{\underline{interval}} | \ (u, v \in D_{\underline{interval}} \wedge u \neq v)$
$\Rightarrow$ *disjoint*$(u, v) \wedge \neg$ *adjacent*$(u, v)\}$

We use $\underline{instant}$ to denote the data type for time instant, which is based on the $\underline{real}$ type. The value of a time interval is to define a set of time instants. Type $\underline{periods}$ represents a set of disjoint and non-connected time intervals. We also give the $\underline{intime}$ type constructor which yields types whose values consist of a time instant and a value. Let α denote an abstract type (excluding time type), e.g., $\underline{int}$, $\underline{real}$.

Definition 3.3 $D_{\underline{intime}} = D_{\underline{instant}} \times D_\alpha$

In addition, we employ spatial and temporal data types from [14, 20]: $\underline{point}$, $\underline{points}$, $\underline{line}$ and $\underline{region}$,$\underline{mbool}$ (moving bool) and $\underline{mpoint}$ (moving point). The definitions of $\underline{mbool}$ and $\underline{mpoint}$ are given in App. C.

[2] Using the algebraic terminology that for a data type α, its domain or carrier set is denoted as D_α.

3.2 Framework

3.2.1 Space and Infrastructures

We let the space for moving objects be covered by the following infrastructures: Free Space, Road Network, Public Transportation Network, Region-based Outdoor and Indoor. Each infrastructure consists of a set of IFOBs and defines available places for moving. For example, roads and streets constitute Road Network, and a set of polygons specifies the walking area for Region-based Outdoor. For Free Space, the object set is empty. Let $I = \{I_{fs}, I_{rn}, I_{ptn}, I_{rbo}, I_{indoor}\}$ be the set of all infrastructures, and $Dom(I_i)(I_i \in I)$ be the values (IFOBs) for I_i. Then, we define the *space* as follows:

Definition 3.4 $Space = \bigcup_{I_i \in I} Dom(I_i)$

Let $\underline{space}$ be the data type representing a space with domain: $D_{\underline{space}} = Space$. Table 1 lists the components for space as well as possible transportation modes in each component. We summarize all transportation modes in Def. 3.5.

$Space$	I_{fs}	*Free*
	I_{rn}	*Car, Bike, Taxi*
	I_{ptn}	*Bus, Train, Metro*
	I_{rbo}	*Walk*
	I_{indoor}	*Indoor*

Table 1: Space Components

Definition 3.5 *Transportation Mode*
$D_{\underline{tm}}$ = {*Car, Bus, Train, Walk, Indoor, Metro, Taxi, Bike, Free*}

Since people also walk in the indoor space, in the following the mode *Walk* means outdoor environment by default, which is to distinguish between the modes *Walk* and *Indoor*. Here, we merely deal with transportation modes for objects moving on the ground. Of course, there are still two modes: *Ship* and *Airplane*. Normally people do not frequently change to these two cases in daily life so that we do not consider them in this paper. But they can be easily integrated where both are considered as the modes in a public transportation system.

3.2.2 Infrastructure Components

The components for an infrastructure are objects representing background geographic information. To manage all these objects in a database system, some data types are needed. For instance, in I_{rn} a line is used to describe the geometrical property of a road, and in I_{rbo} a polygon or a region (in the following, we use terms polygon and region interchangeably) is used to identify a pavement area. The overall IFOBs include not only spatial objects but also spatial-temporal objects such as buses, trains. The data type depends on the infrastructure characteristic. To have a general representation, we give the definition of an IFOB as below.

Definition 3.6 *Infrastructure Object (IFOB)*
An IFOB is defined as $w(oid, s, \beta, name)$ *where* $oid(\in D_{\underline{int}})$ *is a unique identifier,* s *is a symbol for a data type,* β *is a value of the type, and a string describing the name.*

The meaning of s is clear for I_{rn}, I_{rbo} and I_{fs} where $s \in$ IOSymbol = {LINE, REGION, FREESPACE}. The data types $\underline{line}$ and $\underline{region}$ are already defined in spatial databases. For free space without IFOBs, we let the symbol be FREESPACE. Regarding I_{ptn}, the components are routes and moving vehicles that have special features. Taking the bus network as an example, a bus route cannot be simply represented by a line because

the data of bus stops is missing. When modeling bus trips, the route and schedule should be considered. In I_{indoor}, a building consists of a set of rooms and the object representing a room should contain such two pieces of information (1) 2D area and (2) height above the ground level [3] to be able to uniquely identify a location. New data types are designed for I_{ptn} and I_{indoor}, defined in Sec. 4.

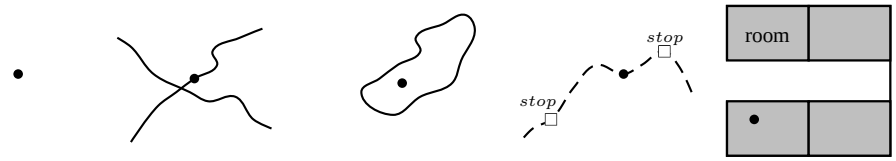
3.2.3 Location Representation

To achieve the goal of general and precise location representation in all environments, in the proposed model we describe the location of a moving object by two parts, defined as follows.

Definition 3.7 *Generic Location*
$D_{\underline{genloc}} = \{(oid, (loc_1, loc_2))|oid \in D_{\underline{int}}, loc_1, loc_2 \in D_{\underline{real}}\}$

The first part is an identifier corresponding to an IFOB and the second stands for the relative position according to that object, described by (loc_1, loc_2). Each IFOB has its own geographical information such as a line for a road, a polygon for a pavement. By accessing the referenced object, the range of the location is determined. With the second value, a global and precise location can be obtained. Using this method, the location in all environments listed in Table 1 can be represented. We explain the semantic meaning of generic location for each infrastructure in the following and give examples in Fig. 1.

- $(\perp, (loc_1, loc_2))$ maps to a location in free space. As I_{fs} does not contain any IFOB, we set oid by $\perp$ and represent the position by recording coordinates (loc_1, loc_2).
- A road network location is represented by $(oid, (loc_1, \perp)$ where oid records a road id and loc_1 describes the location on the road. The value of loc_1 is between zero and the length of the road. loc_2 is not required and set as undefined.
- Regarding Region-based Outdoor, we denote a location $(oid, (loc_1, loc_2))$ by (1) a polygon id and (2) the relative location inside the polygon, with the left lower point of the polygon bounding box as origin point.
- In a public transportation system, a location $(oid, (loc_1, loc_2))$ is specified by a route id, a stop number loc_1 and the distance loc_2 from the stop on the route.
- For an indoor location, we let oid map to a room and (loc_1, loc_2) record the location inside the room. As I_{indoor} is a 3D environment, the height above the ground level of a location needs to be determined. This value is recorded by an IFOB representing a room. All locations inside one room have the same height as the room, so the value does not have to be explicitly recorded. The concept of room is general, e.g., an office room, a corridor.



(a) I_{fs} (b) I_{rn} (c) I_{rbo} (d) I_{ptn} (e) I_{indoor}

Figure 1: Location in Each Infrastructure

To sum up, the proposed method represents the location in all environments. The representation first maps to an IFOB and second identifies the relative position according to that object, resulting in a precise location in

[3] the value means the distance from the road surface after the building construction

space. The underlying geographical data such as line or polygon is recorded by the referenced object and obtained by accessing the context, yielding the capability of supporting both global and local coordinates. In addition, topological relations like "*contains*" and "*inside*" between the location and IFOBs can be directly derived by investigating the object identifier instead of involving costly geometric computation.

Next, we define a data type called $\underline{genrange}$ representing sets of locations. Such a type is used on the one hand to represent the trajectory of a generic moving object, i.e., its projection into generic space. On the other hand, it can represent any kind of curve or region in the generic space, for example, a road in the road network, a park in the region based outdoor space, a collection of rooms in a building, or an arbitrary region in free space.

Definition 3.8 *Generic Range*

Subrange = $\{(oid, l, m) | oid \in D_{\underline{int}}, l \in D_{\underline{points}} \cup D_{\underline{line}} \cup D_{\underline{region}} \cup \{\perp\},$
$m \in D_{\underline{tm}} \cup \{\perp\}\}$
$D_{\underline{genrange}} = 2^{Subrange}$

Each $\underline{genrange}$ value is a set of elements each of which is denoted by sub_traj_i ($\in Subrange$). sub_traj_i is composed of three attributes: oid denotes an IFOB id, l stores a set of locations and m describes a transportation mode.

When a $\underline{genrange}$ value is used to represent the trajectory of a moving object, then l contains the locations passed by the object (usually as a $\underline{line}$ value with coordinates relative to the given IFOB). Also the transportation mode is defined. For example, the trajectory of a car moving on the road.

When the value is used to represent an arbitrary region in the generic space, l may be a $\underline{points}$ or $\underline{region}$ value or undefined (an undefined value means that all locations in the IFOB are valid). In this case the transportation mode may be undefined.

3.3 Moving Objects Representation

In this paper, we focus on the complete history movement of moving objects instead of current and future positions. First, we give the abstract representation.

Definition 3.9 *Let f be the location function projecting from time to location.*

$$f : D_{\underline{instant}} \rightarrow D_{\underline{genloc}}$$

To represent the continuously changing data, a standard way to model it is by the regression method that first segments the data into pieces or regular intervals within which it exhibits a well-defined trend, and then chooses the basis and mathematical functions most appropriate to fit the data in each piece [14, 16, 30, 48]. Using the method of *sliced representation*, we represent a moving object as a set of so-called *temporal units (slices)*. Each unit defines a time interval as well as the movement during the interval.

Definition 3.10 *Generic Temporal Units*

Loc = $\{(loc_1, loc_2) | loc_1, loc_2 \in D_{real}\}$
Gentu = $\{(i, oid, i_{loc_1}, i_{loc_2}, m) | i \in D_{\underline{interval}}, oid \in D_{\underline{int}}, i_{loc_1}, i_{loc_2} \in$ *Loc*, $m \in D_{\underline{tm}}\}$

Each element in Gentu has five components: i defines a time interval, oid maps to an IFOB, i_{loc_1}, i_{loc_2} identify two positions according to the IFOB and m describes the transportation mode. A unit defines that during the time interval i (1) locations are represented by referencing to an IFOB identified by oid; (2) i_{loc_1} (i_{loc_2}) precisely records the start (end) location at $i.s$ ($i.e$) according to the IFOB; (3) the transportation mode is m. Positions during $[i.s, i.e]$ are achieved by linear interpolation.

Given two units $u_1, u_2 \in$ Gentu, some relationships can be exploited between them:

1. $u_1.m \neq u_2.m$

   This denotes two units with different transportation modes. For example, $u_1.m$ = *Car* and $u_2.m$ = *Walk*. Queries on transportation modes extract data from this attribute.

2. $u_1.m = u_2.m \wedge u_1.oid \neq u_2.oid$

   Transportation modes are the same, but the referenced IFOBs are different. Suppose that $u_1.m$ = $u_2.m$ = *Bus*, then $u_1.oid$ and $u_2.oid$ denote different buses.

3. $u_1.m = u_2.m \wedge u_1.oid = u_2.oid$

   In this case, both transportation modes and IFOBs are the same, but the precise locations in space can be different, distinguished by i_{loc_1} and i_{loc_2}. For example, two pedestrians walk on the same pavement or two clerks walk inside the same office room, resulting in different trajectories.

To make a compact representation for moving objects, we define the *mergeable* condition for two units u_1 and u_2. Let $p_1(\in D_{\underline{point}})$ be the start location of u_1. In fact, p_1 is taken from the projection in space of u_1. Again, we can have p_2 for u_2. Since a unit represents a linear movement, a function is employed for the evaluation at time t. We define T_1 (T_2) to be the relative time intervals of u_1 (u_2), i.e., $[0, u_1.i.e\text{-}u_1.i.s]$, and give the evaluation functions.

$f_{u_1}(t) = \{(x, y)|x = p_1.x + a_1 \cdot t, y = p_1.y + b_1 \cdot t, t \in T_1, a_1, b_1 \in \mathbb{R}\}$
$f_{u_2}(t) = \{(x, y)|x = p_2.x + a_2 \cdot t, y = p_2.y + b_2 \cdot t, t \in T_2, a_2, b_2 \in \mathbb{R}\}$
Then we define u_1 and u_2 to be *mergeable* if and only if the following three conditions hold:
(i) $u_1.oid$ = $u_2.oid \wedge u_1.m$ = $u_2.m$;
(ii) $u_1.i$ is *adjacent* to $u_2.i$;
(iii) $f_{u_1}(T_1.e)$ = $f_{u_2}(T_2.s) \wedge a_1 = a_2 \wedge b_1 = b_2$.

A set of sub movements can be merged and compressed into one unit if they fulfill the *mergeable* condition. Suppose that a car is moving on the highway with constant speed for half an hour, a GPS device may record the position every five seconds. Instead of maintaining a large number of GPS tracks one unit can suffice. A generic moving object is defined as a sequence of generic temporal units. The formal definition is given below.

Definition 3.11 *Generic Moving Objects*
$D_{\underline{genmo}}$ = $\{< u_1, u_2, ..., u_n > |n \geq 0, n \in D_{\underline{int}}$, *and*
(i) $\forall i \in [1, n], u_i \in$ *Gentu*
(ii) $\forall i, j \in [1, n], i \neq j \Rightarrow u_i.i \cap u_j.i = \oslash \wedge u_i, u_j$ *are not mergeable* $\}$

3.4 Approximate Location

In some cases, a rough location can simplify the representation and at the same time still be available for the application (e.g., [35, 36]). For example,

Q_3:"*find all travelers passing areas A and B during their trips*".

The data should be managed in such a way that one can determine whether the trajectory of a traveler intersects the area or not, while the precise movement inside can be ignored. Accurate data need much storage space and processing time. Our model supports both precise and imprecise representation so that the system is able to tune the level of scale to the appropriate value, making therefore the system much more flexible. We define the location in an approximate way by $(oid, (\perp, \perp))$. As each IFOB covers some places, the range for the location can be known by accessing the object. Considering *Bobby's* movement M_1, one can roughly describe such a trip by recording a sequence of IFOB ids for: (1) polygons; (2) roads; (3) rooms.

4 Representation for Infrastructures

4.1 Public Transportation Network

Public transportation vehicles include buses, trains, and underground trains. As they have similar characteristics, without loss of generality we take bus network as an example to present how the infrastructure is represented. A bus network contains static and dynamic components, where the first one includes bus stops and bus routes and the second one includes a set of bus trips.

4.1.1 Static

A bus stop identifies a location where a bus arrives, lets passengers get on and off, and then departs (if it is not the last stop). Each stop belongs to a route and several stops from different routes may map to the same location where a transfer can occur.

Bus Stop: Intuitively, a bus stop corresponds to a point in space, but it also has some other information, e.g., a route id. Instead of simply using a point we define a data type named $\underline{busstop}$ with the carrier set:

Definition 4.1 *Bus Stop*
$D_{\underline{busstop}} = \{(rid, pos) | rid, pos \in D_{\underline{int}}, rid \geq 0 \wedge pos \geq 0\}$

To identify a bus stop, we use rid to denote the route id and pos to show the order on the route. Given bs_1, $bs_2 \in D_{\underline{busstop}}$, we define

bs_1 is *adjacent* to $bs_2 \Leftrightarrow bs_1.rid = bs_2.rid \wedge bs_1.pos + 1 = bs_2.pos$.

Bus Route: A bus route consists of a sequence of sub lines (partitioned by bus stops) each of which is called a *segment*. A segment represents the connection between two *adjacent* bus stops.

Definition 4.2 *Bus Segment*
Busseg = $\{(bs_1, bs_2, geo) | bs_1, bs_2 \in D_{\underline{busstop}}$, bs_1, bs_2 *are adjacent,* $geo \in D_{\underline{line}}\}$

Given two segments seg_1, $seg_2 \in$ Busseg, they are called *linkable* if and only if $seg_1.bs_2$ and $seg_2.bs_1$ map to the same point. Let **getbr_id**(seg) return the route id of a segment and $\underline{busroute}$ be the data type for bus routes with the domain:

Definition 4.3 *Bus Route*
$D_{\underline{busroute}}$ = $\{< seg_1, seg_2, ..., seg_n > | n \geq 1, n \in D_{\underline{int}}$, *and*
(1) $\forall i \in [1, n], seg_i \in$ *Busseg;*
(2) $\forall i \in [1, n$ - 1], ***getbr_id***(seg_i) = ***getbr_id***$(seg_{i+1}) \wedge seg_i$ *is linkable to* seg_{i+1} $\}$

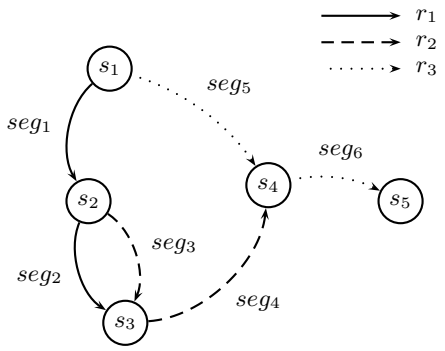


Figure 2: A Simple Bus Network

To demonstrate how the above data types work together, Figure 2 gives an example with three bus routes $\{r_1, r_2, r_3\}$ and five locations $\{s_1, s_2, s_3, s_4, s_5\}$. The location set is to identify the positions for bus stops. Figure 3 lists the representation of bus stops and bus routes. For brevity, we omit the detailed geometry description here. Each location from $\{s_1, s_2, s_3, s_4\}$ shows an intersection between two bus routes, implying that a bus change can happen here.

One issue that needs to be discussed is that in the real world a bus route has two directions (*Up* and *Down*) and for each direction there is a sequence of bus stops. *Up* and *Down* routes are normally located on two different lanes and the geometry description for them should be different. To be consistent with the reality, we define a bus route with *Up* and *Down* as two different routes, and each of them is uniquely identified and represented. In our model, this can be done by setting different $rids$ to denote *Up* and *Down* routes.

Loc	Bus Stops
s_1	$bs_1 =< 1,1 >, bs_7 =< 3,1 >$
s_2	$bs_2 =< 1,2 >, bs_4 =< 2,1 >$
s_3	$bs_3 =< 1,3 >, bs_5 =< 2,2 >$
s_4	$bs_6 =< 2,3 >, bs_8 =< 3,2 >$
s_5	$bs_9 =< 3,3 >$

(a)

Id	Bus Routes
1	$br_1 = \{seg_1(bs_1, bs_2), seg_2(bs_2, bs_3)\}$
2	$br_2 = \{seg_3(bs_4, bs_5), seg_4(bs_5, bs_6)\}$
3	$br_3 = \{seg_5(bs_7, bs_8), seg_6(bs_8, bs_9)\}$

(b)

Figure 3: Static Component

4.1.2 Dynamic

A bus trip is determined by (1) bus route and (2) schedule where the former defines the path and the latter specifies the time period of such a moving object. We model each bus trip as a moving point. Let $D_{\underline{busloc}}$ be the domain of bus locations.

Definition 4.4 *Bus Location*
$D_{\underline{busloc}}$ = $\{(br\_id, bs\_id, pos)\ |br\_id, bs\_id \in D_{\underline{int}}, pos \in D_{\underline{real}}\}$

A bus location is represented by a route id, a stop number on the route and the relative distance from the stop. Note that the definition is consistent with Def. 3.7 in Sec. 3.2.3, and here it is specified as the location on a bus route. Given $bloc_1, bloc_2 \in D_{\underline{busloc}}$, they are called *successive* if and only if the following conditions hold:
(1) $bloc_1.br\_id$ = $bloc_2.br\_id$;
(2) $bloc_1.bs\_id + 1$ = $bloc_2.bs\_id$;
(3) $bloc_1.pos$ = $bloc_2.pos$ = 0.
We define an operator called **geodata** to return the spatial point for a bus location and a bus stop.
geodata: $\underline{busroute} \times \underline{busloc} \rightarrow \underline{point}$
$\underline{busroute} \times \underline{busstop} \rightarrow \underline{point}$
Let $p\_bs_i\ (\in D_{\underline{point}})$ be the point that $bloc_i\ (\in D_{\underline{busloc}})$ corresponds to. Then, the equality of two bus locations is defined as: $bloc_1 = bloc_2 \Leftrightarrow p\_bs_1 = p\_bs_2$. Let BusU be the domain for bus trip units, defined in the following.

Definition 4.5 *Bus Trip Units*
BusU = $\{(i, bloc_1, bloc_2) | i \in D_{\underline{interval}}$, $bloc_1, bloc_2 \in D_{\underline{busloc}}$, *and*
(i) $i.s = i.e \Rightarrow bloc_1 = bloc_2$;
(ii) $bloc_1, bloc_2$ *are successive or equal* $\}$

The definition applies to the framework of generic temporal units in Def. 3.10, Sec. 3.3. Let $v \in$ Gentu, $u \in$ BusU, and we have
(1) $v.i \rightarrow u.i$;
(2) $v.oid \rightarrow u.bloc_1.br\_id\ (u.bloc_2.br\_id)$;
(3) $v.i_{loc_1} \rightarrow (u.bloc_1.bs\_id, u.bloc_1.pos)$;
(4) $v.i_{loc_2} \rightarrow (u.bloc_2.bs\_id, u.bloc_2.pos)$;
(5) as u denotes a specific infrastructure unit, $v.m$ (*Bus*) is omitted in u.
Let $\underline{mpptn}$ be the data type representing *bus trips*. Based on Def. 4.5, we have:

Definition 4.6 *Bus Trips*
$D_{\underline{mpptn}} = \{< tub_1, tub_2, ..., tub_n > | n \geq 1, n \in int$, *and*
(i) $\forall i \in [1, n], tub_i \in$ *BusU;*
(ii) $\forall i, j \in [1, n], i \neq j \Rightarrow tub_i.i \cap tub_j.i = \emptyset$;
(iii) $\forall i, j \in [1, n], tub_i.bloc_1.br\_id = tub_j.bloc_1.br\_id;$ $\}$

Condition (iii) ensures that each bus trip only belongs to one route. A bus moves along the route represented by a curve in space, and positions between two *successive* stops are determined by the offset distances from the starting point on the route. This method yields a compact representation. Compared with the times of frequently updating raw location data (e.g., coordinates), the number of bus stops is very small. As a result, both update and storage costs are considerably reduced.

4.1.3 Infrastructure Objects

With the above data types, we specify the IFOBs representation in I_{ptn}. First, we extend IOSymbol to include values for data types in I_{ptn}, IOSymbol = IOSymbol $\cup$ {BUSSTOP, BUSROUTE, MPPTN [4]}. Applying Def. 3.6 in Sec. 3.2.2, the result is:

- static: IFOB(oid, *BUSSTOP*, β, $name$)($\beta \in D_{\underline{busstop}}$)
  IFOB(oid, *BUSROUTE*, β, $name$)($\beta \in D_{\underline{busroute}}$)
- dynamic: IFOB(oid, *MPPTN*, β, $name$)($\beta \in D_{\underline{mpptn}}$)

Static objects are referenced by dynamic objects, while the movement of humans is represented by referencing to dynamic IFOBs. Here one has to be a bit careful. Although one may be tempted to assume that specifying the time interval and a reference to the bus object in the infrastructure are sufficient to define the movement of a traveler on a bus, a bus trip describes the *scheduled bus movement* from which the real bus movement may deviate (e.g., due to delays). For example, one would like to determine in a query at which bus stop a passenger entered the bus. Deriving this from the scheduled location of the bus at the time the passenger enters may lead to errors and inconsistencies with the remaining trip of the passenger (e.g., in the region based outdoor infrastructure). We therefore include in the unit describing a passenger's trip also the start and end locations on the respective bus route.

Let $u_{ptn}(i, oid, i_{loc_1}, i_{loc_2}, m)$ represent the movement of a bus traveler where:
(1) $oid \rightarrow$ IFOB (oid, *MPPTN*, β, $name$)($\beta \in D_{\underline{mpptn}}$);
(2) $i_{loc_1} \rightarrow (bs\_id, pos)$;
(3) $i_{loc_2} \rightarrow (bs\_id, pos)$;
(4) m = *Bus*.

The unit illustrates that during i the traveler goes by a bus identified by oid from i_{loc_1} to i_{loc_2}, where i_{loc_1} (i_{loc_2}) records the start (end) location, i.e., the bus stop. In detail, bs_id records the stop number and pos denotes the distance from the stop. This representation can significantly reduce the storage size for moving objects. Instead of recording the locations at all places where the bus speed or direction changes, one unit can suffice. If several passengers take the same bus, their locations all map to the same IFOB. Travelers may get on and off the same bus at different stops, which are distinguished by i_{loc_1} and i_{loc_2}. In addition, one does not have to update the data until the travelers get off the bus or switch to another one.

A bus trajectory is a set of elements sub_traj_i ($\in$ Subrange) with the attribute values:
(1) $sub\_traj_i.oid \rightarrow$ IFOB(oid, *BUSROUTE*, β, $name$)($\beta \in D_{\underline{busroute}}$);
(2) $sub\_traj_i.l$ stores the movement on the route;
(3) $sub\_traj_i.m$ = *Bus*.

4.2 Indoor

4.2.1 Modeling Indoor Space

In this environment, IFOBs represent objects such as rooms, chambers, corridors, etc. We use the term $groom$ (general room) for all of them. Each groom covers a 2D area and is located at some distance above the ground level.

[4] MPPTN stands for moving point for public transportation network.

Definition 4.7 *Region3d*

Region3d $=\{(poly, h) | poly \in D_{\underline{region}}, h \in D_{\underline{real}}\}$

The attribute $poly$ describes the 2D area and h denotes the room height above the ground level. Given r_1, r_2 $\in$ Region3d, we define $r_1 = r_2 \Leftrightarrow r_1.poly = r_2.poly \wedge r_1.h = r_2.h$. In a building, usually an office room or a corridor has only one flat surface. But there are also amphitheaters and chambers that have several surfaces with different altitudes. To be more general, we model a groom as a set of objects. See below.

Definition 4.8 *General Room*

$D_{\underline{groom}} = \{GR \subseteq$ *Region3d* $| (gr_1, gr_2 \in GR \wedge gr_1 \neq gr_2) \Rightarrow$
disjoint$(gr_1.poly, gr_2.poly)\}$

Considering the staircase between two floors, such an IFOB is modeled as a set of 3D regions each of which represents one footstep of the staircase. We represent an elevator by several grooms and each object has only one element recording the 2D area on one floor (a rectangle) accompanied with a height value. To clarify terms, when we speak of groom, it is a short description for general room, while $\underline{groom}$ denotes the data type representing a groom. We extend the symbol set IOSymbol to include the value for $\underline{groom}$, IOSymbol = IOSymbol $\cup$ {GROOM}.

Let $u_{indoor}(i, oid, i_{loc_1}, i_{loc_2}, m)$ be a temporal unit for indoor moving objects. Applying Def. 3.10 in Sec.3.3, the attributes map to:

(1) $oid \rightarrow$ IFOB$(oid,$ *GROOM*, $\beta, name)(\beta \in D_{\underline{groom}})$;

(2) $i_{loc_1} \rightarrow (x, y)$;

(3) $i_{loc_2} \rightarrow (x, y)$;

(4) $m =$ *Indoor*.

The coordinates (x, y) denote the relative position in the groom where the left lower point of the bounding box on the 2D area is set as the origin point. An indoor trajectory is specified as:

(1) $sub\_traj_i.oid \rightarrow$ IFOB$(oid,$ *GROOM*, $\beta, name)(\beta \in D_{\underline{groom}})$;

(2) $sub\_traj_i.l$ records the movement inside the IFOB;

(3) $sub\_traj_i.m =$ *Indoor*.

4.2.2 Indoor Navigation

In this part, we present an indoor graph for precise shortest path searching inside a building that contains a set of grooms. Doors are used to build the connection between two grooms, and therefore we first give the representation for doors.

Definition 4.9 *Data Type for Doors*

Doorpos$=\{(gr\_id, pos) | gr\_id \in D_{\underline{int}}, pos \in D_{\underline{line}}\}$

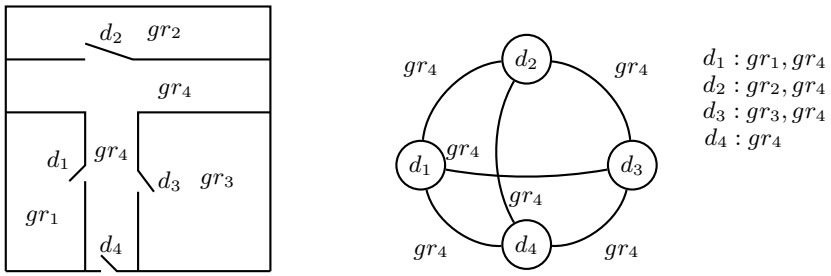
$D_{\underline{door}} = \{(dp_1, dp_2, genus, tp) |$ $dp_1, dp_2 \in$ *Doorpos*, $genus \in \{$*non-elev*, *elev*$\}$,
$tp \in D_{\underline{mbool}}\}$

A door is shared by two grooms. We let dp_1 and dp_2 represent the door position in each groom with the groom id gr_id and the position pos inside. Besides location data, a door has an attribute $genus$ describing the type. To model the time-dependent state of a door, i.e., open or closed, we define a moving bool tp. The value is a sequence of units where each unit has a time interval and a bool. We distinguish between elevator doors (eld) and non-elevator doors (neld). The time-dependent state for neld can be known, while the state for eld is unknown. Usually, from 8am to 6pm an office room door is open, otherwise it is closed. But for an elevator, the floor it currently stays on is unknown. Hence the state for eld is uncertain and tp is set as undefined. We model the entrance/exit of an elevator on each floor as a door which builds the connection between different levels.

Definition 4.10 *An indoor graph is defined as* G_{indoor} $(N, E, W, \sum_{groom}, \sum_{door}, l_v, l_e,)$ *where (1)* N *is a set of nodes; (2)* $E \subseteq V \times V$ *is a set of edges and each edge is associated with a weight value from W; (3)* $l_v : N \rightarrow \sum_{door}$ *is a function assigning labels to nodes; (4)* $l_e : E \rightarrow \sum_{groom}$ *is a function assigning labels to edges.*

In G_{indoor}, we let a node denote a door. An edge corresponds to a groom and builds the connection between two doors without passing through a third door. For example, an elevator is represented by several grooms each of which defines the place for the elevator on one floor. The elevator entrance/exit on each floor is modeled as a node and the connection between two *adjacent* floors shows an edge. One groom may have more than one door and each pair of doors indicates a connection, resulting in several edges in the graph. A shortest path between two doors is computed in the obstructed space for the reason that obstacles may exist inside a room. We store the path on the edge and set the weight as the path length. To create an indoor graph, all paths between doors inside one groom have to be pre-computed. Since a room usually does not have too many doors, the cost is not high. And the computation is done once. We let *indoorgraph* be the data type for indoor graphs.

Fig. 4(a) depicts an example with four rooms $\{gr_1, gr_2, gr_3, gr_4\}$ and four doors $\{d_1, d_2, d_3, d_4\}$. gr_1, gr_2, gr_3 are office rooms and gr_4 is a hallway. Each office room has a door, d_1 in gr_1, d_2 in gr_2 and d_3 in gr_3. d_4 is the entrance/exit for the hallway. G_{indoor} is shown in Fig.4(b). For simplicity, we omit the shortest path for each edge.



(a) (b)

Figure 4: Floor Plan and Indoor Graph

Our graph supports searching a shortest path with different costs: (1) distance; (2) number of rooms; (3) time.

shortest distance: We apply Dijkstra's algorithm on G_{indoor} to find a route with the shortest distance. Let i_{loc} ($\in$ D_{genloc}) be an indoor location which is represented by a groom id and a position inside the groom. A preprocessing step is required that is to find paths from i_{loc} to all doors in the groom where i_{loc} is located, resulting in the connection to G_{indoor}. A^* algorithm can be applied to improve query processing if the start and end locations are at the same level. This is done by setting the Euclidean distance as the heuristic value. We take the center point of each door (represented by a line) to compute the distance. If the start and end locations are at different levels, the above heuristic value might not be efficient (still correct) as it does not involve any information about staircase and elevator which are critical for movement between different levels. Another optimization technique is developed. The heights above the ground level of start and end locations can be retrieved by accessing their grooms. The two values define the height range for all locations on the shortest path. Then, all doors whose height values are out of the range can be pruned during searching.

smallest number of rooms: This case is simple as the total cost is achieved by aggregating the number of edges in the path.

minimum traveling time: Let v_{walk} be a person's average speed when walking inside a building. Since each edge stores the shortest path between two doors inside a groom, the time for passing an edge can be calculated by v_{walk} and the path length. This method applies for all grooms except elevators. The time cost of moving by an elevator contains two parts: (1) waiting; (2) elevator moving. The cost of the first part is uncertain because on

which floor the elevator is located cannot be determined at a query time instant. The second part can be easily calculated if the height between two floors and the elevator speed is given. To solve the problem, we process as follows. Assume that the elevator speed is a constant value. Let d_i^{elev} (d_j^{elev}) denote the elevator door on the ith (jth) floor. The whole time needed to move from d_i^{elev} to d_j^{elev} depends on (i) arrival time at d_i^{elev} and (ii) the path along which the elevator moves to reach d_j^{elev}. Without loss of generality, we assume $0 \leq i < j$. In the best case, when a person arrives at d_i^{elev}, the elevator happens to stay at floor i and directly moves up to floor j. On the contrary, the elevator might just leave from floor i and moves up so that the person has to wait until the elevator goes down to the bottom floor and moves up again. Suppose that the distance between two floors is the same for the whole building, denoted by h, and there are n floors in total. Then, the length of the shortest and longest path between floor i and j is $(j-i)*h$ and $(j-i)*h+2*n*h$, respectively. The two values are the lower and upper bounds, and the set of all possible values is $\{(j-i)*h, (j-i)*h+h, \ldots, (j-i)*h+2*n*h\}$. Each value can be assigned a membership probability depending on the elevator schedule. A simple solution is to impose the uniform distribution of the probability for each path length, while more realistic modeling should take into account the building structure, history data analysis, optimal elevator schedule, and so on. Consequently, the total time spent on the elevator can be computed.

We compare our indoor model with some others [31, 25, 57] in the literature. (1) The precise indoor location and shortest path are represented by our method. Existing techniques only locate the object inside a room, while the accurate position inside is not handled. Previous indoor graphs define a room as a node, resulting an approximate description for indoor shortest path, i.e., a sequence of rooms. In some cases, this might not provide enough information for a traveler as some buildings may have large rooms with complex structures and obstacles, e.g., the hall in a hotel or an airport. We model the precise area of each room and define the paths to establish the connections between doors. This can have the exact location of an indoor object and show a well-defined route for people to follow. (2) We model both spatial and temporal attributes for doors to have a practical and robust representation, instead of only illustrating the connections between rooms. As a result, the concept of doors in our model is general, not only for office rooms and corridors but also for staircases and elevators. Previous models do not concern about the time-dependent state of a door. (3) We design an indoor graph to support optimal routes searching with respect to different costs all of which are meaningful in practice. Previous techniques do not provide a precise indoor shortest path and do not model the time cost of an indoor trip.

4.3 Region-based Outdoor

This environment is for people walking outdoor, including places such as pavements, zebra crossings, etc. We represent the whole area by a relatively large polygon P with many holes inside. Holes denote obstacles such as building blocks and junction areas which people cannot directly pass through. To efficiently manage the data, P is decomposed into a set of polygons (e.g., triangles) to be stored in the database. Then, a location is represented by (1) a polygon id; (2) the relative position inside the polygon where the origin point is the left lower point of the polygon bounding box. Let $u_{rbo}(i, oid, i_{loc_1}, i_{loc_2}, m)$ denote a unit for moving objects by walking and the value of each attribute is specified as: (1) $oid \rightarrow$ IFOB(oid, *REGION*, β, $name$)($\beta \in D_{\underline{region}}$); (2) $i_{loc_1} \rightarrow (x, y)$; (3) $i_{loc_2} \rightarrow (x, y)$; (4) m = *Walk*. Positions between i_{loc_1} and i_{loc_2} during i are obtained by linear interpolation. Fig. 5 shows an example in which the areas drawn by crosshatching represent the places that people cannot pass through. $\{pl_1,...,pl_7\}$ denote the subareas from decomposing P. An example movement is depicted, denoted by M_3, passing $\{pl_1, pl_5, pl_7\}$. Locations of such a moving object can be represented in an approximate way (Sec. 3.4) where only the referenced object id is recorded, $M_3 = <(1), (5), (7)>$ (for brevity, we ignore other values of a unit).

4.4 Free Space and Road Network

Location representation in previous models [14, 20, 18] is seamlessly integrated into the proposed framework. Let $u_{fs}(i, oid, i_{loc_1}, i_{loc_2}, m)$ be a unit for moving objects in free space. Specifically, (1) $oid \rightarrow \perp$; (2) $i_{loc_1} \rightarrow (loc_1, loc_2)$; (3) $i_{loc_2} \rightarrow (loc_1, loc_2)$; (4) m = *Free*. As there is no IFOB in I_{fs}, oid is undefined. (loc_1, loc_2)

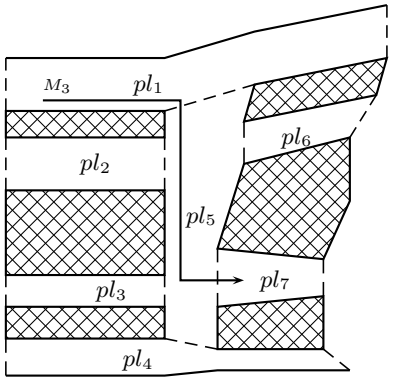


Figure 5: Outdoor Space Partition

denotes the coordinates in space. Let $u_{rn}(i, oid, i_{loc_1}, i_{loc_2}, m)$ be a unit for moving objects in a road network where (1) $oid \rightarrow$ IFOB $(oid, \textit{LINE}, \beta, name)(\beta \in D_{\underline{line}})$; (2) $i_{loc_1} \rightarrow (loc_1, \perp)$; (3) $i_{loc_2} \rightarrow (loc_1, \perp)$; (4) $m \in \{\textit{Car}, \textit{Taxi}, \textit{Bicycle}\}$. A network position is represented by a road id and the position on the road. The second attribute in i_{loc_1} (i_{loc_2}) is set as undefined. In both environments, positions between i_{loc_1} and i_{loc_2} in a unit are computed by linear interpolation.

5 The Type System and An Interface

5.1 Data Types

The proposed data types, summarized in Table 2, include generic types for all environments and those arising from some specific infrastructures developed in the previous sections. We give the type system in Table 3 [5] where SPATIAL and GRAPH [20, 18] are still used in one infrastructure. Generic data types $\underline{genloc}$, $\underline{genrange}$ and $\underline{genmo}$ are available in all cases. For example, $\underline{genloc}$ represents the location in any infrastructure and is specified as $\underline{point}$ in free space and $\underline{gpoint}$ in road network. The type $\underline{mpptn}$ that we define for moving buses is an instance of $\underline{genmo}$ in I_{ptn}. Similarly, moving objects representation in I_{fs} and I_{rn} is also an instance of $\underline{genmo}$. To clarify the difference between the new model and previous models, we give the type system of free space and road network in App. C.

	Name	**Meaning**
generic data types	$\underline{tm}$	transportation modes
	$\underline{genloc}$	generic locations
	$\underline{genrange}$	generic sets of locations
	$\underline{genmo}$	generic moving objects
PTN	$\underline{busstop}$	bus stops
	$\underline{busroute}$	bus routes
	$\underline{busloc}$	locations on bus routes
	$\underline{mpptn}$	moving buses
Indoor	$\underline{groom}$	general rooms
	$\underline{door}$	doors

Table 2: A Summary of Proposed Data Types

[5] basic types such as $\underline{int}$, $\underline{bool}$ are omitted

	→ SPATIAL	*point*, *points*, *line*, *region*
	→ GRAPH	*qpoint*, *gline*
	→ PTN	*busstop*, *busroute*, *busloc*
	→ INDOOR	*groom*, *door*
	→ GENLOC	*genloc*
	→ SPACE	*genrange*, *space*
	→ TM	*tm*
GENLOC	→ TEMPORAL	*moving*, *intime*

Table 3: The Type System in General Model

5.2 Space: A Relational View

We provide an interface to exchange information between values of proposed data types and a relational environment. First, all IFOBs are managed in the database system as they are referenced by moving objects. Table 4 shows the defined infrastructures each of which may have several components. For each component, we create a relation where each tuple corresponds to an IFOB. Each respective relation schema is shown in Table 5.

Infrastructure	Infrastructure Component
I_{ptn}	BUSSTOP BUSROUTE BUS
I_{indoor}	ROOM DOOR ROOMPATH
I_{rbo}	OUTDOOR
I_{rn}	ROAD
I_{fs}	

Table 4: Infrastructures and Their Components

$rel_{busstop}$	`(BusStopId: int, Stop: busstop, Name: string)`
$rel_{busroute}$	`(BusRouteId: int, Route: busroute, Name: string, Up: bool)`
rel_{bus}	`(BusId: int, BusTrip: mpptn, Name: string)`
rel_{room}	`(RoomId: int, Room: groom, Name: string)`
rel_{door}	`(DoorId: int, Door: door)`
$rel_{roompath}$	`(RoomPathId: int, Door1: int, Door2: int, Weight: real, Room: groom, Name: string, Path: line)`
rel_{rbo}	`(RegId: int, Reg: region, Name: string)`
rel_{rn}	`(RoadId: int, Road: line, Name: string)`

Table 5: Infrastructure Relations

The data type to define an IFOB is embedded as an attribute in each relation. To have a unique identifier for each IFOB, we define a range of *int* values to denote IFOB ids for each infrastructure and different infrastructures are assigned disjoint values. With all infrastructure relations, we construct the space in two steps:

1. create a space initialized by one relation:

createspace: $\underline{rel} \rightarrow \underline{space}$

2. add more infrastructures to the space:

   put_infra: $\underline{space} \times \underline{rel} \rightarrow \underline{space}$

In the first step, the input relation can be empty, then we have the free space. If the relation stores roads, we have the space with road network. In the second step, more infrastructures can be added. One can create a full or non-full space depending on the application. For example, a non-full space might be road network plus bus network. Afterwards, the infrastructure data can be accessed by:

get_infra: $\underline{space} \times \underline{int} \rightarrow \underline{rel}$

The value of the second argument is from the set {BUSSTOP, BUSROUTE, BUS, ROOM, DOOR, ROOMPATH, OUTDOOR, ROAD}, i.e., the names of infrastructure components from Table 4, whose elements are assumed to be available as integer constants. Suppose that we have infrastructure relations for a city called Gendon. Applying the two steps (**createspace** and **put_infra**) we create a full space, denoted by SpaceGendon. The following examples illustrate queries on infrastructure data.

- **Q1**. "Show me the information of *Alexander* street."

```
SELECT *
FROM get_infra(SpaceGendon, ROAD) as road
WHERE road.Name = "Alexander"
```

- **Q2**. "Where can I switch between bus No.12 and No.37?"

```
SELECT bs1, bs2
FROM get_infra(SpaceGendon, BUSSTOP) as bs1,
     get_infra(SpaceGendon, BUSSTOP) as bs2,
     get_infra(SpaceGendon, BUSROUTE) as br1,
     get_infra(SpaceGendon, BUSROUTE) as br2

WHERE br1.BusRouteId = 12 AND br2.BusRouteId = 37 AND
      bs1.Stop.rid = 12 AND bs2.Stop.rid = 37 AND
      geodata(br1,bs1) = geodata(br2,bs2)
```

To answer **Q2**, one needs to perform a join on two relations: bus routes and stops. **geodata** is defined in Sec.4.1.2, returning the location of a bus stop.

Assume we also have some trajectory data of citizens living and working in Gendon. A trip is represented by a tuple recording the trip id, trajectory and name. The relation is named MOGendon and has the schema:

```
MOGendon(Mo_id: int, Traj: genmo, Name: string)
```

5.3 Graph Model for Indoor Navigation

Recall that in Sec. 4.2.2 we define nodes by doors and edges by paths between doors inside one groom for an indoor graph. We use two relations to store graph nodes and edges, rel_{door} for doors and $rel_{roompath}$ for paths. The relation schemas are included in Table 5. An indoor graph is created by the following operator.

createindoorgraph: $\underline{rel} \times \underline{rel} \rightarrow \underline{indoorgraph}$

Then, we can run shortest path queries using

indoornavigation: $\underline{genloc} \times \underline{genloc} \times \underline{instant} \times \underline{int} \times \underline{indoorgraph} \rightarrow \underline{genrange}$

where the first two arguments specify the start and end locations, the third indicates a query time and the fourth argument of type $\underline{int}$ denotes the cost type of such a shortest path (e.g., distance, time).

6 Operations

Before proposing operators and query examples, we first introduce some notations employed from [20, 18] to achieve a smooth integration with abstract data types and their operations. The command `LET <name> = <query>` creates a new database object `name` whose value is given by the query expression. A query can thus be formulated in several steps (separated by `;`), defining intermediate results by `LET`. The last expression determines the result of the query. We define expressions for time instant and interval, e.g., **instant**(2010, 12, 5, 8) (8am on Dec. 5, 2010), **interval**((2010, 12, 5, 8), (2010, 12, 5, 9)) (between 8am and 9am on Dec.5,2010). To define operator semantics, some notations are needed. Considering Def. 3.6, a space can be denoted by

$Space$ = $\{(oid\ ,\ s,\ \beta,\ name) | oid \in D_{\underline{int}},\ s \in$ IOSymbol, $name \in string\}$.

By default, $Space$ is available for all operators, i.e., not needed as an explicit argument, and one element of $Space$ is denoted by w. We let u, v be single values of a data type and correspondingly U, V be generic sets of values of a type. u (U) refers to the first argument and v (V) refers to the second argument in an operator. mo denotes a moving object and u_i be one unit of mo. Semantics of some operators is given in this section while that of the others is put into App. A due to complex definitions.

6.1 Extended Operators

It would not make sense to start from scratch here. The design considers aspects such as systematic construction of the type system, definition of generic operations ranging over large collections of data types, and consistency between non-temporal and temporal (i.e. time dependent) operations. Operators in this part have the same semantics in previous work [20, 18] for free space and road network, listed in Table 6. We extend the syntax in order to support generic data types. See Q3.

Name	Signature
$=, \neq$	$\underline{tm} \times \underline{tm} \rightarrow \underline{bool}$
deftime	$\underline{genmo} \rightarrow \underline{periods}$
duration	$\underline{periods} \rightarrow \underline{real}$
present	$\underline{genmo} \times \underline{intime} \rightarrow \underline{bool}$
	$\underline{genmo} \times \underline{periods} \rightarrow \underline{bool}$
initial, final	$\underline{genmo} \rightarrow \underline{intime}(\underline{genloc})$
atinstant	$\underline{genmo} \times \underline{instant} \rightarrow \underline{intime}(\underline{genloc})$
atperiods	$\underline{genmo} \times \underline{periods} \rightarrow \underline{genmo}$
val	$\underline{intime}(\underline{genloc}) \rightarrow \underline{genloc}$
inst	$\underline{intime}(\underline{genloc}) \rightarrow \underline{instant}$

Table 6: Operators by Extending Syntax

- **Q3**. Where is *Bobby* at 8:00 am?

```
LET qt = instant(2010, 12, 5, 8);

SELECT val(mo.Traj atinstant qt)
FROM MOGendon AS mo
WHERE mo.Name = "Bobby"
```

The result is expressed by a value with $\underline{genloc}$, denoted by gl. One can investigate $gl.oid$ and $Space$ to know the infrastructure gl is located. Assuming the infrastructure is Road Network, the referenced road can be retrieved.

```
SELECT *
FROM get_infra(Space, ROAD) as r
WHERE r.RoadId = gl.oid
```

6.2 Spatial and Spatial-Temporal

Spatial operators are collected in Table 7. We give comments for some operators. The meaning for **distance** should be clear if the two arguments belong to the same infrastructure, e.g., Euclidean distance in I_{fs}, network distance in I_{rn}. If they belong to different infrastructures, we define the value to be the minimum length of a trip connecting the locations.

Table 8 lists operators on spatial-temporal data types. **Trajectory** projects a moving object into the space. Given a location, **at** restricts the trip to the specified place. One can also restrict the movement to a set of places by giving a $\underline{genrange}$ argument. The operator **trip** takes two locations and a query instant as input. The locations are general (i.e., can be situated in any infrastructure), and the result is described in the form of $\underline{genmo}$. Consider the query "*find a trip from my office room to my home with minimum traveling time*". The resulting trip described by a sequence of transportation modes might be *Indoor* $\rightarrow$ *Car* $\rightarrow$ *Walk* or *Indoor* $\rightarrow$ *Walk* $\rightarrow$ *Bus* $\rightarrow$ *Walk*.

We give query examples for these operators below.

Name	Signature	Semantics
$=, \neq$	$\underline{genloc} \times \underline{genloc} \rightarrow \underline{bool}$	Def. 10.1 in App. A
inside	$\underline{genloc} \times \underline{genrange} \rightarrow \underline{bool}$	Def. 10.2 in App. A
intersects	$\underline{genloc} \times \underline{genrange} \rightarrow \underline{bool}$	the same as Def. 10.2
	$\underline{genrange} \times \underline{genrange} \rightarrow \underline{bool}$	Def. 10.3 in App. A
distance	$\underline{genloc} \times \underline{genloc} \rightarrow \underline{real}$	Def. 10.4 in App. A

Table 7: Spatial Operators

Name	Signature	Semantics
trajectory	$\underline{genmo} \rightarrow \underline{genrange}$	Def. 10.5 in App. A
at	$\underline{genmo} \times \underline{genloc} \rightarrow \underline{genmo}$	Def. 10.6 in App. A
	$\underline{genmo} \times \underline{genrange} \rightarrow \underline{genmo}$	Def. 10.8 in App. A
passes	$\underline{genmo} \times \underline{genloc} \rightarrow \underline{bool}$	**at**$(mo, v) \neq \emptyset$
	$\underline{genmo} \times \underline{genrange} \rightarrow \underline{bool}$	**at**$(mo, v) \neq \emptyset$
trip	$\underline{genloc} \times \underline{genloc} \times \underline{instant} \rightarrow \underline{genmo}$	omitted

Table 8: Spatial-Temporal Operators

- **Q4**. Between 8am and 9am, who sits in the same bus as *Bobby*?

```
LET qt = interval((2010, 12, 5, 8), (2010, 12, 5, 9));

SELECT mo1.Name
FROM MOGendon AS mo1, MOGendon AS mo2
WHERE  mo2.Name = "Bobby" AND
       val(mo1.Traj atperiods qt) =
       val(mo2.Traj atperiods qt)
```

6.3 Sets and Decomposition

The design of types and operations in the abstract model for moving objects [20] emphasized compatibility with a relational model and therefore proposed only "atomic" data types, i.e., types suitable as attribute types in a relation. Presumably for this reason, there is no $\underline{set}$ type constructor applicable to atomic types so that for example, a type $\underline{set}(\underline{periods})$ would be available.

However, it was recognized that a tool for decomposing values of a data type into components was needed. For example, for a $\underline{region}$ value consisting of several disjoint components ("faces") it should be possible to obtain each face as an independent $\underline{region}$ value; similarly for a $\underline{moving}(\underline{point})$ one would like to have a decomposition into continuous pieces, each as a separate $\underline{mpoint}$ value. It is obvious that a natural operation to perform such decompositions would have a signature

components: $\alpha \rightarrow \underline{set}(\alpha)$

for any atomic type α consisting of several components. In this paper we leave this restriction behind and do introduce a $\underline{set}$ constructor. In the implementation, it has turned out that one does not even need an explicit data structure to represent such sets but can handle sets of values as streams of tuples at the executable level of the system. Of course, it is also possible to introduce an explicit data structure for sets. Together with the set constructor we define a few generic operations:

contains: $\underline{set}(\alpha) \times \alpha \rightarrow \underline{bool}$
card: $\underline{set}(\alpha) \rightarrow \underline{int}$

Furthermore, the operation **components** is offered for decomposition:

For $\alpha \in \{\underline{range}(\beta), \underline{points}, \underline{line}, \underline{region}, \underline{moving}(\gamma)\}$:
components: $\alpha \rightarrow \underline{set}(\alpha)$

Note that type $\underline{periods}$ is just an abbreviation for $\underline{range}(\underline{instant})$, hence the signature

components: $\underline{periods} \rightarrow \underline{set}(\underline{periods})$

is also available.

6.4 Transportation Modes and IFOBs

Name	Signature	Semantics
get_mode	$\underline{genmo} \rightarrow \underline{set}(\underline{tm})$	$\{u_i.m \mid u_i \in mo\}$
at	$\underline{genmo} \times \underline{tm} \rightarrow \underline{genmo}$	Def. 6.2
ref_id	$\underline{ioref} \rightarrow \underline{int}$	$u.oid$
	$\underline{busroute} \rightarrow \underline{int}$	$u.bs_1.rid$, $u \in U$
	$\underline{mpptn} \rightarrow \underline{int}$	$u_i.bloc_1.br\_id$, $u_i \in mo$
get_ref	$\underline{genloc} \rightarrow \underline{ioref}$	Def. 6.3
	$\underline{genrange} \rightarrow \underline{set}(\underline{ioref})$	Def. 6.4
	$\underline{genmo} \rightarrow \underline{set}(\underline{ioref})$	Def. 6.5

Table 9: Operators on Transportation Modes and Infrastructure Objects

Table 9 lists the proposed operators. Given a moving object, we can get its transportation modes. Using *Bobby*'s trajectory M_1 as an example, **get_mode** returns {*Walk*, *Car*, *Indoor*}. With **get_mode** and **contains** (Sec. 6.3), one can examine whether a moving object includes a specific transportation mode. See **Q7**.

- **Q5**. Find all people using public transportation vehicles.

```
SELECT mo.Name
FROM MOGendon AS mo
WHERE get_mode(mo.Traj) contains Bus OR
      get_mode(mo.Traj) contains Train OR
      get_mode(mo.Traj) contains Metro
```

Given a transportation mode, a trip can be restricted to a sub movement with respect to the mode, done by **at** (in Sec. 6.2, **at** restricts a trip to a given space and now the operator is extended).

- **Q6**. How long does *Bobby* walk during his trip?

```
SELECT duration(deftime(mo.Traj at Walk))
FROM MOGendon AS mo
WHERE mo.Name = "Bobby"
```

We define a data type named $\underline{ioref}$ to have a light representation of referenced IFOBs, whose value may need a large storage space, e.g., region.

Definition 6.1 *Reference Data Type*
$D_{\underline{ioref}}$ = $\{(oid, ref)|oid \in D_{\underline{int}}, ref \in$ *IOSymbol*$\}$

Operator **ref_id** returns the referenced object id and **get_ref** gets the IFOB in a light representation (See **Q9**). When the underlying data is needed, one can get the full representation by accessing $Space$.

- **Q7**. Find all people taking "Bus527".

```
SELECT mo.Name
FROM MOGendon AS mo,
     get_infra(SpaceGendon, BUS) AS bus
WHERE ref_id(bus.BusTrip) = 527 AND
      get_ref(mo.Traj at Bus) contains  bus.BusId
```

Definition 6.2 $\underline{genmo} \times \underline{tm} \rightarrow \underline{genmo}$ ***at***
The result is $< u_1, u_2,...,u_n >$ *where* $u_i.m = v$.

Definition 6.3 $\underline{genloc} \rightarrow \underline{ioref}$ ***get_ref***
The result is $(u.oid, w.s)$ *where* $\exists\, w \in Space$*:* $u.oid = w.oid$.

Definition 6.4 $\underline{genrange} \rightarrow \underline{set}(\underline{ioref})$ ***get_ref***
The result is $\{(u.oid, w.s) \mid u \in U \wedge (\exists\, w \in Space : u.oid = w.oid)\}$.

Definition 6.5 $\underline{genmo} \rightarrow \underline{set}(\underline{ioref})$ ***get_ref***
The result is $\{(u_i.oid, w.s) \mid u_i \in mo \wedge (\exists\, w \in Space : u_i.oid = w.oid)\}$.

6.5 Subtype Relationships and Conversions Between Generic and Specific Types

Sometimes one needs to apply generic operations (defined for $\underline{genloc}$, $\underline{genrange}$ or $\underline{genmo}$) to objects of more specific types. This is possible through subtype relationships. In Section 4, data types for the different infrastructures have been shown to fit into the generic framework; hence arguments of the specific types can be substituted in operations for the generic types. The valid subtype relationships are shown in Table 10. We give an example **Q10** that is between $\underline{genloc}$ and $\underline{busstop}$.

- **Q8**. How long does *Bobby* wait at the bus stop "Uni"?

Infrastructure	Generic Types		
	genloc	*genrange*	*genmo*
I_{ptn}	*busstop*, *busloc*	*busroute*	*mpptn*
I_{indoor}		*groom*, *door*	
I_{rbo}			
I_{rn}	*gpoint*	*gline*	*mgpoint*
I_{fs}	*point*	*points*, *line*, *region*	*mpoint*

Table 10: Subtype Relationships

```
SELECT duration(deftime((mo.Traj at Walk) at bs.Stop))
FROM MOGendon AS mo,
     get_infra(SpaceGendon, BUSSTOP) AS bs
WHERE mo.Name = "Bobby" AND
      bs.Name = "Uni"
```

We give some comments for the query. The movement is first restricted to the mode *Walk*, and then limited to a bus stop. We suppose that people walk to a bus stop instead of by car or some other modes. Here, a generic location is specified as the bus stop location. **deftime** gets the time period at the place and **duration** returns the time span.

Furthermore, sometimes it is necessary to compare locations or moving objects that belong to different infrastructures. To be able to evaluate such relationships, we might try to provide mappings between all pairs of infrastructures. However, it is simpler to introduce a generic mapping that converts an object from any infrastructure into the free space counterpart. We introduce such an operation called **freespace**. Essentially it maps values of any of the types in Table 10 to the corresponding type in free space, hence offers the signatures shown in Table 11. For the mapping of indoor locations into free space we ignore the height value associated with a *groom* and project into the (x, y) plane (or the ground level of the building). This maps rooms from different floors into the same locations. Nevertheless, the mapping is still able to relate locations between different infrastructures. To help understand the operators, see the examples below.

Operator	Signature
freespace	*genloc* $\rightarrow$*point*
	busstop $\rightarrow$*point*
	busloc $\rightarrow$*point*
	gpoint $\rightarrow$*point*
	busroute $\rightarrow$*line*
	groom $\rightarrow$*region*
	door $\rightarrow$*line*
	gline $\rightarrow$*line*
	genmo $\rightarrow$*mpoint*
	mpptn $\rightarrow$*mpoint*
	mgpoint $\rightarrow$*mpoint*
freespace_p	*genrange* $\rightarrow$*points*
freespace_l	*genrange* $\rightarrow$*line*
freespace_r	*genrange* $\rightarrow$*region*
genloc	*int* $\times$*real* $\times$*real* $\rightarrow$*genloc*

Table 11: Conversion Operators

- **Q9**. Is the university under a thunderstorm area?

  Let R_storm ($\in D_{\underline{region}}$) be the thunderstorm area.

```
SELECT R_storm contains freespace(room.Room),
FROM get_infra(SpaceGendon, ROOM) AS room,
WHERE room.Name contains "Uni"
```

We assume the name of all university rooms has ”Uni“ as the prefix and project the area of a room to the free space.

- **Q10**. Find all buses passing the city center area.

```
SELECT bus.Name
FROM get_infra(SpaceGendon, BUS) AS bus,
     get_infra(SpaceGendon, OUTDOOR) AS outdoor
WHERE outdoor.Name = "CityCenter" AND
      freespace(bus.Bus) passes outdoor.Reg
```

To be able to compare the bus movement (of type $\underline{mpptn}$) with the city center region (which is a $\underline{region}$ value in free space) we must map it into the free space using operation **freespace**, i.e., convert it to an $\underline{mpoint}$.

- **Q11**. Did bus 527 pass any traveler going by bicycle?

  We define *pass* to mean that there exists a time instant where the distance between the two moving objects is less than 3 meters.

```
SELECT mo.Name
FROM get_infra(SpaceGendon, MPPTN) AS bus, MOGendon AS mo
WHERE ref_id(bus.BusTrip) = 527 AND
      sometimes(distance(freespace(bus.Vehicle),
                         freespace(mo.Traj at Bicycle)) < 3.0)
```

Here we need to determine the distance between two moving objects, namely, the bus and a traveler going by bicycle. We assume the bus passes the bicycle if at some time their distance is small enough. To be able to determine the time-dependent distance, both moving objects have to move within the same infrastructure. Therefore we first map both of them into free space, i.e., convert them to $\underline{mpoint}$. The application of the **distance** operator returns a moving real. Comparing this with the constant 3.0 returns an $\underline{mbool}$ value. Finally, **sometimes** yields true, if the $\underline{mbool}$ ever assumes the value true.[6]

For the mapping of $\underline{genrange}$ values into free space, we need to introduce three operators **freespace_p**, **freespace_l**, and **freespace_r** returning $\underline{points}$, $\underline{line}$, or $\underline{region}$ values, respectively. This is necessary because $\underline{genrange}$ is a union type. The operators return the parts of the argument that can be mapped into the respective result type. For example, one can obtain the trajectory of a trip as a $\underline{line}$ value.

Finally, for querying, a construction operator **genloc** for generic locations is needed that builds a generic location from an IFOB identifier and two real “coordinates”.

[6] **sometimes** is a derived operation, **sometimes**(mb) = **not**(**isempty**(**deftime**(mb **at** true))). See [21], Exercise 4.5.

7 More Query Examples

In this section, we perform queries to evaluate the data model. The notation `SET(<name>, <value>)` constructs a relation with a single tuple and attribute from an atomic value. For example, `SET(Name, "Bobby")` constructs the relation r of the previous paragraph. Seeing the operator **components** (Sec. 6.3) with the signature $\underline{periods} \rightarrow \underline{set}(\underline{periods})$, the notation `SET(Time, components(p))` will produce a relation with schema `(Time: periods)` having one tuple for each distinct time interval in the $\underline{periods}$ value p.

Appendix-A provides useful information to check all query formulations in detail. It lists the schemas of the involved infrastructure relations and an index for each query showing where the used operations have been defined.

- **Q12**. "Which streets does bus12 pass by."

```
SELECT r
FROM get_infra(SpaceGendon, ROAD) as r,
     get_infra(SpaceGendon, BUSROUTE) as br
WHERE br.BusRouteId = 12 AND
      intersects(br.Route, r.Road)
```

- **Q13**. Find out who passed the room r_{312} in the university between 8am and 9am.

```
LET qt = interval((2010, 12, 5, 8), (2010, 12, 5, 9));
```

Assume the name of the room is "Uni-312".

```
SELECT mo.Name
FROM MOGendon AS mo,
     get_infra(SpaceGendon, ROOM) AS room
WHERE room.Name = "Uni-312" AND
      get_ref((mo.Traj atperiods qt) at Indoor)
      contains room.room_id
```

To define this query, several operators are needed. First, we restrict moving objects to the given period by **atperiods** and to indoor movement by **at**. Second, we get the referenced IFOBs by **get_ref**. The result is represented by the reference type, i.e., as $\underline{set}(\underline{ioref})$. Third, we check whether the room id is included by **contains**.

- **Q14**. Where does *Bobby* walk during his trip?

```
SELECT trajectory(mo.Traj at Walk)
FROM MOGendon AS mo
WHERE mo.Name = "Bobby"
```

- **Q15**. Find all people staying at office room 312 at the university for more than 2 hours.

```
SELECT mo.Name
FROM MOGendon AS mo, get_infra(SpaceGendon, ROOM) AS room
WHERE room.Name = "Uni-312" AND EXISTS
  SELECT *
  FROM SET(Piece,
    components(deftime(mo.Traj at room.Room)))
  WHERE duration(Piece) > 120
```

The trajectory of mo is restricted to the times when he/she was at room 312 which due to the subtype relationship can be used as a *genrange* value. Obviously, the query refers to a single stay at this office rather than to the aggregated time over many visits. Hence we decompose the definition time interval into disjoint intervals using **components**. This is transformed into a relation from which we select tuples for which the duration of the stay is more than two hours. If the relation is not empty for a given mo, then this mo qualifies for the result.

- **Q16**. Who entered bus 527 at bus stop "University"?

```
SELECT mo.Name
FROM MOGendon AS mo,
     get_infra(SpaceGendon, BUS) AS bus,
     get_infra(SpaceGendon, BUSSTOP) AS busstop
WHERE ref_id(bus.BusTrip) = 527 AND
      busstop.Name = "University" AND
      bus.Stop = val(initial(mo.Traj at
                          genloc(bus.BusId, undef, undef)))
```

- **Q17**. Did anyone who was at the University on floor H-2 between 4:30pm and 5pm take a bus to the main (train) station?

  To be on floor H-2 means to be in any of the rooms of floor H-2. We assume a table is available associating rooms with the floors they belong to:

```
Uni_Rel(Floor: string, RoomName: string)
```

  Then the query can be formulated as follows:

```
LET qt1 = interval((2010, 12, 5, 16, 30), (2010, 12, 5, 17));
LET qt2 = interval((2010, 12, 5, 16, 30), (2010, 12, 5, 19));

SELECT mo.Name
FROM MOGendon AS mo, Uni_Rel AS u,
     get_infra(SpaceGendon, ROOM) AS r,
     get_infra(SpaceGendon, BUSSTOP) AS bs1,
     get_infra(SpaceGendon, BUSSTOP) AS bs2
WHERE u.Floor = "H-2" AND
      u.RoomName = r.Name AND
      (mo.Traj atperiods qt1) passes r.Room AND
      bs1.Name = "University" AND
      bs2.Name = "Main station" AND
      val(initial((mo.Traj atperiods qt2) at Bus))
      = bs1.Stop AND
      val(final((mo.Traj atperiods qt2) at Bus))
      = bs2.Stop
```

The query implicitly requires that the person takes the bus to the main station later when he is on floor H-2. Hence we define a second time period from 4:30pm to 7pm during which the bus must be taken. We then ask for trajectories of people passing through the generic locations of rooms on floor H-2 within period $qt1$ as well as having a bus trip during $qt2$ that starts at the University bus stop and ends at the main station bus stop.

- **Q18**. Who arrived by taxi at the university in December?

  To arrive by taxi at the university means that the final location of the passenger within the taxi belongs to some driveway area close to the university. We assume such a part of the road network has been entered into the database as an object UniDriveway of type *gline*.

```
LET December = interval((2010, 12, 1), (2010, 12, 31));

SELECT mo.Name
FROM MOGendon AS mo
WHERE EXISTS
  SELECT *
  FROM SET(Trip,
    components((mo.Traj atperiods December) at Taxi))
  WHERE val(final(Trip)) inside UniDriveway
```

We reduce the trajectory of a person to the taxi trips in December. There may be several such trips (present in the resulting trajectory), hence we apply **components** to get the continuous pieces, i.e., the individual taxi trips. In the end, we check whether the relation containing these trips includes one with the final destination within the university driveway area.

8 Implementation

8.1 Data Management

The implementation is done in an open source and extensible database system SECONDO [17] that supports spatial and moving objects management. The proposed data model is built in three steps.

First, five infrastructures (Road Network, Region-based Outdoor, Bus Network, Metro Network and Indoor) are developed and each infrastructure contains four components: the type, a set storing its IFOBs, an index set and a graph. All data types representing IFOBs are implemented and each of them is embedded as an attribute in a relation. In order to access these objects efficiently, several indices are built. B-trees are built on relations by specifying key values and r-trees are also maintained to efficiently support queries on spatial attributes. Sometimes, one may compare objects from different infrastructures. For example, given a building, all bus stops within 300 meters can be located. R-tree indices improve the query efficiency. To sum up, an infrastructure organizes a set of relations and indices. Using the road network as an example, a collection of roads is stored in a relation, a B-tree with the key road id and an R-tree are built on the relation, and a road graph is defined for searching the shortest path.

Second, the space which is built on the top of all infrastructures stores a set of records. Each record references to an infrastructure. Concrete data about IFOBs are maintained by the low level infrastructure and loaded by the space if they are needed for query processing (operator **get_infra** in Section 5.2). A record has two values denoting the minimum and maximum object ids in an infrastructure to efficiently identify the environment for a location. That is, the semantic meaning of oid (Def. 3.7) is explained by the space, i.e., a road or room id. An infrastructure can run individually without depending on other infrastructures to have a flexible system, as some applications may not need all environments. In general, the space serves as an interface and builds the connection between underlying infrastructure objects and moving objects. During the query processing, concrete data are loaded from the low level infrastructure, e.g., roads, bus routes, rooms. Only a reference for each infrastructure is maintained by the space to have a light representation. Besides, the space has the functionality of location mapping. To compare infrastructure objects from different environments, (e.g., **Q10**, **Q11**) one needs to project them into the same system and this is done by the space.

Third, generic data types that can apply in all infrastructures are developed. A value represented by a generic type can be converted to a specific representation in one infrastructure, e.g., $\underline{genloc} \rightarrow \underline{point}$. All proposed operators are implemented and registered as functions in the DBMS in order to be available for query users.

8.2 Data Generator

To evaluate the performance of such a system, a benchmark is needed that provides realistic data including both infrastructures and moving objects. Motivated by the lack of real data for all infrastructures in a consistent

environment, we create outdoor infrastructures from real roads and indoor environment from public floor plans (e.g., [7, 5]) to simulate a city environment. [55] demonstrates the created data including roads, pavement areas, bus routes, buildings, etc. Visualizing 3D indoor objects such as rooms, staircases, indoor trajectories is available by implementing a 3D viewer in the system. One can also perform the animation of indoor moving objects. We let the infrastructure data size be fixed and do not consider any update, e.g., roads insertion, new building construction.

Moving objects are created based on trip planning where the start and end locations may be situated in different infrastructures, yielding multiple transportation modes. A navigation algorithm through all environments is developed to support queries like "*find the shortest path from the office room to my apartment*". The system generates a moving object with different transportation modes, e.g., *Walk* $\rightarrow$ *Bus* $\rightarrow$ *Walk* $\rightarrow$ *Indoor*. To find such a path, different infrastructure graphs are developed such as pavement graph and bus graph. We briefly describe the procedure of creating the example movement. First, given a query location in the walking area, the nearest bus stop is found and mapped to a pavement location. Then, the shortest path from the query location to the bus stop is returned and such a path is located in the pavement area. Second, the procedure of routing in a bus network is called and a bus trip with the minimum traveling time is retrieved. From the ending bus stop, the traveler walks to the target building. Third, indoor navigation is called to find a precise shortest path from the building entrance to a room. In addition to producing moving objects with multiple transportation modes, trips in a single environment are also available such as walking in the pavement area and moving inside a building. Precise paths are used to produce these moving objects instead of approximate locations.

8.3 Indexing

To optimize the query processing, an index structure is built on each generic moving object to efficiently access the units. Recall that a moving object is represented by a set of units arranged in a linear order on time. For the purpose of obtaining a sub trip according to the transportation mode in a fast way, we employ an index to access the data rather than search all units linearly. See some examples in the proposed queries that request a sub trip according to the mode. To answer **Q4** the movement with the mode *Bus* is returned, and in **Q6** walking trips are specified. For travelers who take public vehicles, one might get the sub trip on a particular bus, seeing **Q7** and **Q16**. Obviously, the sequential scan yields poor performance for large data.

Given a generic moving object mo, let $I = \{(m, l, h) | m \in D_{\underline{tm}}, l, h \in D_{\underline{int}}\}$ be the index built on the units of mo. Each element in I consists of three attributes where m records the transportation mode and l, h are entries pointing to the start and end locations of a sequence of units with the mode m. That is, each element maps to a sub movement in mo with a single mode. Consider such an example movement [7]

$mo = <$ (*Indoor*)$_1$, (*Indoor*)$_2$, ... , (*Indoor*)$_{10}$, (*Walk*)$_{11}$, (*Walk*)$_{12}$, ... , (*Walk*)$_{15}$,
(*Car*)$_{16}$, (*Car*)$_{17}$, ..., (*Car*)$_{30}$, (*Walk*)$_{31}$, ... , (*Walk*)$_{40}$,
(*Indoor*)$_{41}$, (*Indoor*)$_{42}$, ..., (*Indoor*)$_{45}>$.

The index built on mo is I = {(*Indoor*, 1, 10), (*Walk*, 11, 15), (*Car*, 16, 30), (*Walk*, 31, 40), (*Indoor*, 41, 45)}. Although I is still a list structure, the quantity of elements depends on the number of modes in mo, usually quite few. Each element locates the range of continuous units with a certain mode. As a result, given a transportation mode we first scan the index to determine the positions of qualified units and then access them to get the concrete value. The searching performance can be improved as a large number of units that do not fulfill the condition can be skipped by accessing I.

By observation, there is a frequently called procedure in the queries, checking the existence of a transportation mode in mo. Since a large part of queries specifies a certain transportation mode (e.g., **Q6**, **Q7**, **Q11**), one needs to find all qualified moving objects. Trips that do not contain the give mode should not be considered. To know the involved transportation modes for mo, one option is to sequentially scan all units and compare the mode attribute to see if the value is identical to the argument. The proposed index can accelerate the procedure, but still yields a linear searching. The ability to determine the existence of a mode in constant time can reduce the overall

[7] for simplicity, we only show the transportation mode in each unit

running time. The solution is as follows. An integer IM (32 bits) is assigned to each mo to denote the involved transportation modes where a bit indicates whether a mode exists or not. Each transportation mode is assigned a value as an index to locate the corresponding bit in IM. We mark the bit *true* if the mode exists and *false* if it does not exist. Suppose that the least significant (right-most) nine bits are used for the modes in Def. 3.5. We assign 0 for *Car* as the bit index, 1 for *Bus*, and so on. A moving object with the following sequence of modes *Indoor* $\rightarrow$ *Walk* $\rightarrow$ *Bus* $\rightarrow$ *Walk* defines the value 26 (binary 00011010). We calculate IM for each mo in advance and store such a value in a tuple as an attribute along with mo. During the query processing, both the query mode and the integer are taken as input. The index for the bit corresponding to the query mode is calculated, and the stored integer is examined.

9 Experimental Evaluation

In this section, we perform the experimental evaluation to test the system that manages generic moving objects. The implementation is developed in Secondo [17] and programmed in C/C++ and Java. A standard PC (AMD 3.0 GHz, 4 GB memory, 2TB disk) running Suse Linux (kernel version 11.3) is used. We utilize the tool MWGen [56] to create all infrastructure data based on real roads and public floor plans (e.g., [7, 5, 6]). The road dataset is Berlin [4]. The tool takes roads and floor plans as input and generates pavement areas, bus network, metro network and a set of buildings. A website [3] is published for providing all experimental materials. The system is open source, and implementations as well as datasets are published for experimental repeatability.

9.1 Datasets

All infrastructure data are shown in Figure 6(a) and the detailed information of buildings is reported in Figure 6(b). We create a set of buildings of different types based on their floor plans to simulate a city environment. $|P|$ means the vertex number of the large polygon for walking, $|B|$ records the number of buildings per type and $|R|$ denotes the number of rooms in one building. Due to the privacy problem, there are no floor plans for private houses. Besides the static IFOBs, there are buses and metros represented by moving objects. In both bus and metro networks, we define a schedule for each route to create trips. We simulate one week movement. For the bus schedule, there are more trips on the day $\in \{$Mon, Tue, Wed, Thu, Fri, Sat$\}$ than on Sunday. For metros, the schedules are the same for the whole week. The detailed information is shown in Figure 6(c).

X Range	[0, 44411]
Y Range	[0, 34781]
Roads	3,250
$\|P\|$	116,516
Bus Routes	89
Metro Routes	10
Buildings	4,996
Size	2.5 G

(a) Infrastructures

Type	$\|B\|$	$\|R\|$
house	3,713	
officeA	600	294
officeB	487	214
shopping mall	80	360
cinema	6	21
hotel	39	584
hospital	46	89
university	20	431
train station	1	56

(b) Statistics of Buildings

No. Bus Trips in One Day (From Mon to Sat)	5,220
No. Bus Trips On Sun	2,660
No. Metro Trips in One Day (From Mon to Sun)	1,897

(c) Mobile Infrastructure Data

Figure 6: Infrastructure Data

Moving objects datasets are described in Figure 7. A set of movement rules is defined in [54] to create trips that simulate a variety of real life scenarios. For example, people's trajectories exhibit *regular patterns* [15] most of the time, e.g., commuting. On the weekend, based on the habits and preferences they may have some trips to interesting places such as home of friends, shopping malls, and cinemas. To perform an activity, people usually prefer nearby to distant places, e.g., look for the nearest hotel. Based on these rules, a set of moving objects with different transportation modes is generated. Figure 7(a) lists the involved transportation modes and Figure 7(b)

shows the properties of the moving objects dataset. The mode *Indoor* is optional in a trip and the result depends on the start and end locations. The regular trips between home and work places include such a mode, but trips representing people walking around in the city center do not contain such a value. The mode *Walk* is contained by each trip. According to the study in [59], walk segment is an important part which builds the connection between movement segments with different transportation modes. Usually people do not directly switch from *Car* to *Bus* or from *Indoor* to *Metro*.

[Indoor] + Walk + { *Bike*, *Car*, *Bus*, *Metro*, *Taxi* }

(a) Transportation Modes

Name	Trips No.	Total Units No. (M)	Avg. Units No. Per Trip	Disk Size (G)
Berlin500k	503,759	25.9	51.4	3.92

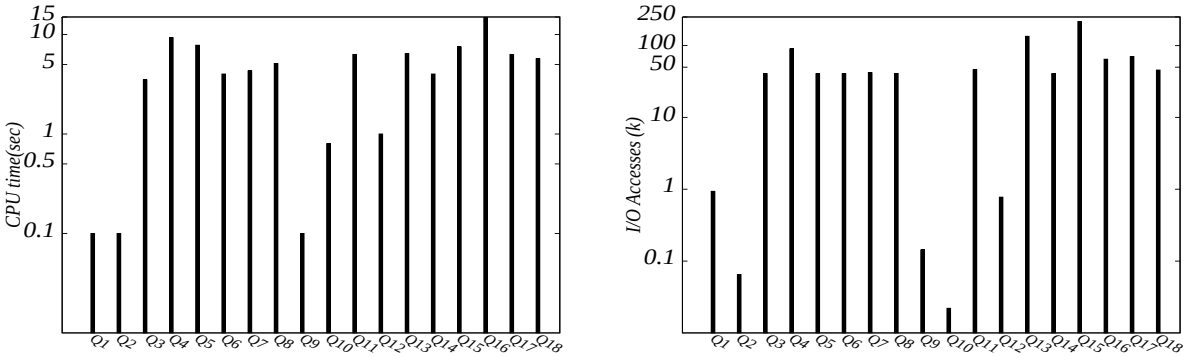
(b) Moving Objects

Figure 7: Generic Moving Objects

9.2 Performance

In this part, we report the experimental results to illustrate the efficiency and effectiveness of the proposed techniques for managing and querying generic moving objects. We investigate the system performance by running queries on a large amount of moving objects with multiple transportation modes. The CPU time and I/O accesses are used as performance metrics where the I/O accesses mean the number of pages that are read into the cache. In the database system, the cache size is set as 64M and one page size is 4k. We run each query ten times and set the average value to be the final result. For the constant value of a query such as a particular person, a university or an office building, we manually select an arbitrary value from the possible set.

Figure 8 reports the experimental results where the CPU time of most queries is less than 10 seconds, demonstrating the efficiency of the proposed approaches. The detailed value of each query is shown in Table 9.



(a) CPU Time (b) I/O Accesses

Figure 8: Query Cost

10 Conclusions and Future Work

In this paper, we introduced a data model to efficiently manage moving objects in multiple environments including public transportation network, indoor, region-based outdoor, road network and free space. A generic location representation is proposed and can be applied in all real world environments. We let the space for moving objects be covered by a set of infrastructures each of which corresponds to an environment. Each infrastructure is composed

Query	CPU Time (sec)	I/O Accesses (k)
Q1	0.1	0.934
Q2	0.1	0.065
Q3	3.5	40.73
Q4	9.3	89.67
Q5	7.8	40.66
Q6	4.0	40.73
Q7	4.3	41.85
Q8	5.1	40.92
Q9	0.1	0.143

(a)

Query	CPU Time (sec)	I/O Accesses (k)
Q10	0.8	0.022
Q11	6.3	45.87
Q12	1.0	0.775
Q13	6.4	134.48
Q14	4.0	40.64
Q15	7.5	215.58
Q16	14.8	64.515
Q17	6.3	70.123
Q18	5.7	45.612

(b)

Figure 9: Experimental Statistics

of a set of objects representing available places for moving objects. The location of moving objects maps to these infrastructure objects. New data types are proposed to define moving objects and components in each infrastructure. To efficiently access the data and formulate queries, a set of operators is defined in the system. A relational interface is provided for exchanging data between values of proposed data types and a relational environment. We formulate a group of interesting queries and report how such a model is implemented in a database system. Experimental results are reported by running queries on a large amount of moving objects.

The future work is to discover some interesting movement patterns from generic moving objects regarding different environments and transportation modes.

References

[1] http://code.google.com/transit/spec/transit_feed_specification.html.

[2] http://conversations.nokia.com/2008/09/23/indoor-positioning-coming-to-life/.

[3] http://dna.fernuni-hagen.de/secondo.html /transportationmode.html.

[4] http://www.bbbike.de/cgi-bin/bbbike.cgi (2008).

[5] http://www.edenresort.com/home.

[6] http://www.greenhosp.org/floor_plans.asp (2011).

[7] http://www.modulargenius.com/default.aspx.

[8] V. Bauer, J. Gamper, R. Loperfido, S. Profanter, S. Putzer, and I. Timko. Computing isochrones in multi-modal, schedule-based transport networks. In *ACM GIS, Demo*, 2008.

[9] J. Booth, P. Sistla, O. Wolfson, and I.F. Cruz. A data model for trip planning in multimodal transportation systems. In *EDBT*, 2009.

[10] S. Brakatsoulas, D. Pfoser, and N. Tryfona. Modeling, storing and mining moving object databases. In *IDEAS*, 2004.

[11] L. Chen, M.T. Özsu, and V. Oria. Robust and fast similarity search for moving object trajectories. In *SIGMOD*, 2005.

[12] Z. Chen, H.T. Shen, X. Zhou, Y. Zheng, and X. Xie. Searching trajectories by locations - an efficiency study. In *SIGMOD*, 2010.

[13] Z. Ding and R.H. Güting. Managing moving objects on dynamic transportation networks. In *SSDBM*, 2004.

[14] L. Forlizzi, R.H. Güting, E. Nardelli, and M. Schneider. A data model and data structures for moving objects databases. In *SIGMOD*, pages 319–330, 2000.

[15] M. C. González, C. A. Hidalgo R., and A. Barabási. Understanding individual human mobility patterns. *Nature*, 453:779–282, 2008.

[16] S. Grumbach, P. Rigaux, and L. Segoufin. Manipulating interpolated data is easier than you thought. In *VLDB*, 2000.

[17] R.H. Güting, V. Almeida, D. Ansorge, T. Behr, Z. Ding, T. Höse, F. Hoffmann, and M. Spiekermann. Secondo: An extensible dbms platform for research prototyping and teaching. In *ICDE, Demo Paper*, 2005.

[18] R.H. Güting, V.T. de Almeida, and Z.M. Ding. Modeling and querying moving objects in networks. *VLDB Journal*, 2006.

[19] R.H. Güting, T. Behr, and J. Xu. Efficient k-nearest neighbor search on moving object trajectories. *VLDB Journal*, 2010.

[20] R.H. Güting, M.H. Böhlen, M. Erwig, C.S. Jensen, N.A. Lorentzos, M. Schneider, and M. Vazirgiannis. A foundation for representing and querying moving objects. *ACM TODS*, 25(1):1–42, 2000.

[21] R.H. Güting and M. Schneider. *Moving Objects Databases*. Morgan Kaufmann, 2005.

[22] C. Hage, C.S. Jensen, T.B. Pedersen, L. Speicys, and I. Timko. Integrated data management for mobile services in the real world. In *VLDB*, 2003.

[23] G.S. Iwerks, H. Samet, and K. Smith. Continuous k-nearest neighbor queries for continuous moving points with updates. In *VLDB*, 2003.

[24] C.S. Jensen, A. Kligys, T.B. Pedersen, and I. Timko. Multidimensional data modeling for location-based services. *VLDB Journal*, 13:1–21, 2004.

[25] C.S. Jensen, H. Lu, and B. Yang. Graph model based indoor tracking. In *MDM*, 2009.

[26] C.S. Jensen, H. Lu, and B. Yang. Indexing the trajectories of moving objects in symbolic indoor space. In *SSTD*, 2009.

[27] H. Jeung, Q. Liu, H.T. Shen, and X. Zhou. A hybrid prediction model for moving objects. In *ICDE*, 2008.

[28] H. Jeung, M.L. Yiu, X. Zhou, C.S. Jensen, and H.T. Shen. Discovery of convoys in trajectory databases. In *VLDB*, 2008.

[29] B. Kuijpers and W. Othman. Trajectory databases: Data models, uncertainty and complete query languages. In *ICDT*, 2007.

[30] J.A. Lema, L. Forlizzi, R.H. Güting, and M. Schneider. Algorithms for moving objects databases. *The Computer Journal*, 46(6):680–712, 2003.

[31] B. Lorenz, H.J. Ohlbach, and E.P. Stoffel. A hybrid spatial model for representing indoor environments. In *W2GIS*, 2006.

[32] K. Mouratidis, M. Hadjieleftheriou, and D. Papadias. Conceptual partitioning: An efficient method for continuous nearest neighbor monitoring. In *SIGMOD*, 2005.

[33] K. Mouratidis, Y. Lin, and M.L. Yiu. Preference queries in large multi-cost transportation networks. In *ICDE*, 2010.

[34] K. Mouratidis, M.L Yiu, D. Papadias, and N. Mamoulis. Continuous nearest neighbor monitoring in road networks. In *VLDB*, 2006.

[35] C. Mouza and P. Rigaux. Mobility patterns. *GeoInformatica*, 9(4):297–319, 2005.

[36] C. Mouza, P. Rigaux, and M. Scholl. Efficient evaluation of parameterized pattern queries. In *CIKM*, 2005.

[37] R. Praing and M. Schneider. Modeling historical and future movements of spatio-temporal objects in moving objects databases. In *CIKM*, pages 183–192, 2007.

[38] R. Praing and M. Schneider. A universal abstract model for future movements of moving objects. In *AGILE Conf.*, pages 111–120, 2007.

[39] S. Reddy, M. Mun, J. Burke, D. Estrin, M.H. Hansen, and M.B. Srivastava. Using mobile phones to determine transportation modes. *TOSN*, 6(2), 2010.

[40] P. Scarponcini. Generalized model for linear referencing in transportation. *GeoInformatica*, 6(1):35–55, 2002.

[41] S. Shekhar, M. Coyle, B. Goyal, D.R. Liu, and S. Sarkar. Data models in geographic information systems. In *Commun ACM 40*, volume 4, pages 103–111, 1997.

[42] P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao. Modeling and querying moving objects. In *ICDE*, pages 422–432, 1997.

[43] L. Speicys and C.S. Jensen. Enabling location-based services – multi-graph representation of transportation networks. *GeoInformatica*, 2008.

[44] L. Speicys, C.S. Jensen, and A. Kligys. Computational data modeling for network-constrained moving objects. In *ACM-GIS*, 2003.

[45] L. Stenneth, O. Wolfson, P. Yu, and B. Xu. Transportation mode detection using mobile devices and gis information. In *ACM SIGSPATIAL*, 2011.

[46] J. Su, H. Xu, and O.H. Ibarra. Moving objects: Logical relationships and queries. In *SSTD*, 2001.

[47] Y. Tao, D. Papadias, and Q. Shen. Continuous nearest neighbor search. In *VLDB*, 2002.

[48] A. Thiagarajan and S. Madden. Querying continuous functions in a database system. In *SIGMOD*, 2008.

[49] I. Timko and T. B. Pedersen. Capturing complex multidimensional data in location-based data warehouses. In *GIS*, pages 147–156, 2004.

[50] M. Vazirgiannis and O. Wolfson. A spatiotemporal model and language for moving objects on road networks. In *SSTD*, 2001.

[51] A. Voisard and B. David. A database perspective on geospatial data modeling. *TKDE*, 14(2):226–242, 2002.

[52] O. Wolfson, S. Chamberlain, K. Kalpakis, and Y. Yesha. Modeling moving objects for location based services. In *IMWS*, 2001.

[53] O. Wolfson, B. Xu, S. Chamberlain, and L. Jiang. Moving objects databases: Issues and solutions. In *SSDBM*, pages 111–122, 1998.

[54] J. Xu and R. H. Güting. GMOBench: A benchmark for generic moving objects. Informatik-Report 362, Fernuniversität in Hagen, 2012.

[55] J. Xu and R.H. Güting. Infrastructures for research on multimodal moving objects. In *MDM, Demo Paper*, 2011.

[56] J. Xu and R.H. Güting. MWGen: A Mini World Generator. In *MDM, To Appear*, 2012.

[57] B. Yang, H. Lu, and C.S. Jensen. Probabilistic threshold k nearest neighbor queries over moving objects in symbolic indoor space. In *EDBT*, 2010.

[58] J. Zhang, M. Zhu, D. Papadias, Y. Tao, and D.L. Tee. Location-based spatial queries. In *SIGMOD*, 2003.

[59] Y. Zheng, Y. Chen, X. Xie, and W.Y. Ma. Understanding transportation mode based on gps data for web application. *ACM Transaction on the Web*, 4(1):1–36, 2010.

[60] Y. Zheng, L. Liu, L. Wang, and X. Xie. Learning transportation mode from raw gps data for geographic applications on the web. In *WWW*, 2008.

Appendix-A: Operator Semantics

We extend the definition of operator **geodata** and give domains and ranges in Table 12. **geodata** which plays a crucial rule for defining semantics, maps locations in different infrastructures into free space. The first three signatures are for PTN. Given a line and the relative position on the line, **geodata** returns a point for that position. Given a point (line) inside a region, we obtain its global position. The last one maps an indoor location to free space. This is done by ignoring the height above the ground level, i.e., projecting a room to the ground floor of a building. To have the symbol set denoting data types supported by **geodata**, we define IOSymbol' = {LINE, REGION , BUSROUTE , GROOM } ($\subset$ IOSymbol).

geodata	$\underline{busroute} \rightarrow \underline{line}$
	$\underline{busroute} \times \underline{busstop} \rightarrow \underline{point}$
	$\underline{busroute} \times \underline{busloc} \rightarrow \underline{point}$
	$\underline{line} \times \underline{real} \rightarrow \underline{point}$
	For $\alpha \in \{\underline{point}, \underline{line}\}$
	$\underline{region} \times \alpha \rightarrow \alpha$
	$\underline{groom} \times \alpha \rightarrow \alpha$

Table 12: Extension for **geodata**

Definition 10.1 ***=:*** $\underline{genloc} \times \underline{genloc} \rightarrow \underline{bool}$

Let u, v be the two locations and the result is TRUE iff

(i) $u.oid = \perp \wedge v.oid = \perp \wedge u.i_{loc} = v.i_{loc}$*; or*

(ii) $u.oid = v.oid \wedge u.i_{loc} = v.i_{loc}$*; or*

(iii) $u.oid = \perp \wedge (\exists\, w \in Space$*:*
$w.oid = v.oid \wedge w.s \in$ *IOSymbol' $\wedge$*
$u.i_{loc} =$ ***geodata*** $(w.\beta, v.i_{loc}))$*; or*

(iv) $v.oid = \perp \wedge (\exists w \in Space$*:*
$w.oid = u.oid \wedge w.s \in$ *IOSymbol' $\wedge$*
$v.i_{loc} =$ ***geodata*** $(w.\beta, u.i_{loc}))$

Cases (i) and (ii) are straightforward as both locations are in free space or in the same IFOB. In case (iii) and (iv), if one location is in free space and the other belongs to another infrastructure, the latter maps to free space by loading the referenced IFOB.

Definition 10.2 ***inside:*** $\underline{genloc} \times \underline{genrange} \rightarrow \underline{bool}$

Let u be a single location and V be a set of locations. The result is TRUE iff $\exists v \in V$ such that

(i) $u.oid = \perp \wedge v.oid = \perp \wedge u.i_{loc} \in v.l$*; or*

(ii) $u.oid = v.oid \wedge (u.i_{loc} \in v.l \vee$
$(\exists w \in Space$*:* $v.oid = w.oid \wedge$
geodata $(w.\beta, u.i_{loc}) \in$ ***geodata*** $(w.\beta, v.l)))$*; or*

(iii) $u.oid = \perp \wedge (\exists w \in Space$*:*
$v.oid = w.oid \wedge w.s \in$ *IOSymbol' $\wedge$*
$u.i_{loc} \in$ ***geodata*** $(w.\beta, v.l))$*; or*

(iv) $v.oid = \perp \wedge (\exists w \in Space$*:*
$u.oid = w.oid \wedge w.s \in$ *IOSymbol $\wedge$*
geodata $(w.\beta, u.i_{loc}) \in v.l)$

Case (i) shows a single location and a set of locations in free space. If u, v reference to the same object w (case (ii)), one can directly compare the location inside w or load the referenced object to transform from relative

location to global. Similarly, if one parameter corresponds to free space and the other does not, we need to perform location mapping by the referenced object. These are cases (iii) and (iv).

Definition 10.3 ***intersects:*** $\underline{genrange} \times \underline{genrange} \rightarrow \underline{bool}$

The result is TRUE iff $\exists u \in U, \exists v \in V$ *such that*

(i) $u.oid = \perp \wedge v.oid = \perp \wedge u.l$ ***intersects*** $v.l$ *; or*

(ii) $u.oid = v.oid \wedge u.l$ ***intersects*** $v.l$ *; or*

(iii) $u.oid = \perp \wedge (\exists w \in Space:$
$v.oid = w.oid \wedge w.s \in$ *IOSymbol' ∧*
$u.l$ ***intersects geodata*** $(w.\beta, v.l))$*; or*

(iv) $v.oid = \perp \wedge (\exists w \in Space:$
$u.oid = w.oid \wedge w.s \in$ *IOSymbol' ∧*
$v.l$ ***intersects geodata***$(w.\beta, u.l))$

Definition 10.4 ***distance:*** $\underline{genloc} \times \underline{genloc} \rightarrow \underline{real}$

length(***trajectory***(***trip***(u, v)))

Definition 10.5 ***trajectory:*** $\underline{genmo} \rightarrow \underline{genrange}$

The result is a set of (oid, l, m) *where* $m = u_i.m$ *and the values for* oid *and* l *are defined in the following.*

(i) $u_i.oid = \perp \Rightarrow oid = \perp \wedge l = \cup f(t)(t \in u_i.i)$*; or*

(ii) $\exists w \in Space:$ $u.oid = w.oid \wedge$ *(*$w.s$ *= REGION ∨* $w.s$ *= GROOM), then*
$oid = u.oid$*,* $l = \cup f(t).i_{loc}(t \in u_i.i)$*; or*

(iii) $\exists w \in Space:$ $u.oid = w.oid \wedge$ *(*$w.s$ *= LINE ∨* $w.s$ *= BUSROUTE), then*
$oid = u.oid, l = \cup$ ***geodata***$(w.\beta, f(t).i_{loc})(t \in u_i.i)$*; or*

(iv) $\exists w_1, w_2 \in Space:$ $u.oid = w_1.oid \wedge w_1.s =$ *MPPTN ∧* ***ref_id***$(w_1) = w_2.oid$*, then*
$oid = w_2.oid, l =$ ***trajectory(atperiods***$(w_1, u_i.i))$

The trajectory of a moving object is a set of movement projections, each of which shows the path of a unit $u_i \in mo$. The meaning for (i) and (ii) should be clear. In case (iii), the object moves along a pre-defined path, e.g., a road or bus route. For case (iv), the location in u_i maps to a bus. Then, the trajectory is the bus movement, which goes to case (iii).

Definition 10.6 ***at:*** $\underline{genmo} \times \underline{genloc} \rightarrow \underline{genmo}$

Let $mo = < u_1, u_2, ..., u_n >$ *be the moving object and* v *denote the location.*

(i) $v.i_{loc} \neq \perp$*, the result is* $< u'_1,...,u'_k >$ *where* $u'_i \in mo \wedge \forall t \in u'_i.i : f(t) = v$*; or*

(ii) $v.oid \neq \perp \wedge v.i_{loc} = \perp$*, then the result is* $< u'_1,...,u'_k >$ *where* $u'_i \in mo \wedge$

(a) $u'_i.oid = v.oid$*; or*

(b) $u'_i.oid = \perp \wedge (\exists w \in Space$ *such that*
$w.oid = v.oid \wedge \forall t \in u'_i.i : f(t) \in w.\beta \wedge w.s \in$ *IOSymbol')*

The meaning is clear if the second argument represents a precise location (case (i)). If the location only records an object id (case (ii)), then the units in the result could have the same reference id as the input, or the unit location maps to the place covered by the given IFOB.

Definition 10.7 ***intersection:*** $\underline{genrange} \times \underline{genrange} \rightarrow \underline{genrange}$

Let U *and* V *be the two arguments, and the result is denoted by* L*. The value for* L *is defined in the following:* $\forall\, l_i \in D_{\underline{genloc}}$*:* l_i ***inside*** $L \Rightarrow l_i$ ***inside*** $U \wedge l_i$ ***inside*** V*.*

Definition 10.8 ***at:*** $\underline{genmo} \times \underline{genrange} \rightarrow \underline{genmo}$

Let $mo = < u_1, u_2, ..., u_n >$ *be the moving object and* V *be a set of locations. We use* $L\ (\in D_{\underline{genrange}})$ *to denote the intersection locations between* ***trajectory***(mo) *and* V*. The result is* $mo' = < u'_1, u'_2, ... , u'_k >$ *fulfilling the condition:* $\forall u'_j \in mo', \forall t \in u'_j.i : f(t)(\in D_{\underline{genloc}})$ ***inside*** L*.*

Appendix-B: Example Relations and Query Signatures

The following relations are used in the queries throughout the paper. Infrastructure relations are accessed by the **get_infra** operator.

```
MOGendon(Mo_id: int, Traj: genmo, Name: string)

rel_busstop  (BusStopId: int, Stop: busstop, Name: string)
rel_busroute (BusRouteId: int, Route; busroute, Name: string,
               Up: bool)
rel_bus      (BusId: int, BusTrip: mpptn, Name: string)
rel_room     (RoomId: int, Room: groom, Name: string)
rel_door     (DoorId: int, Door: door)
rel_roompath (RoomPathId: int, Door1: int, Door2: int, Weight: real,
               Room: groom, Name: string, Path: line)
rel_rbo      (RegId: int, Reg: region, Name: string)
rel_rn       (RoadId: int, Road: line, Name: string)
```

In the following tables we show the signatures of operations used in the queries. Operator **get_infra** is omitted as it occurs in almost every query. We define subtype to mean that the operator performs a conversion from a specific type to a generic type.

No.	Operator	Signature		Def.
Q1				
Q2	**geodata**	$\underline{busroute} \times \underline{busstop}$	$\rightarrow \underline{point}$	Sec. 4.1.2
Q3	**atinstant**	$\underline{genmo} \times \underline{instant}$	$\rightarrow \underline{intime(genloc)}$	Table 6
	val	$\underline{intime(genloc)}$	$\rightarrow \underline{genloc}$	Table 6
Q4	**atperiods**	$\underline{genmo} \times \underline{periods}$	$\rightarrow \underline{genmo}$	Table 6
	val	$\underline{intime(genloc)}$	$\rightarrow \underline{genloc}$	Table 6
	=	$\underline{genloc} \times \underline{genloc}$	$\rightarrow \underline{bool}$	Table 7
Q5	**get_mode**	$\underline{genmo}$	$\rightarrow \underline{set(tm)}$	Table 9
	contains	$\underline{set(tm)} \times \underline{tm}$	$\rightarrow \underline{bool}$	Sec. 6.3
Q6	**at**	$\underline{genmo} \times \underline{tm}$	$\rightarrow \underline{genmo}$	Table 9
	deftime	$\underline{genmo}$	$\rightarrow \underline{periods}$	Table 6
	duration	$\underline{periods}$	$\rightarrow \underline{real}$	Table 6
Q7	**ref_id**	$\underline{mpptn}$	$\rightarrow \underline{int}$	Table 9
	at	$\underline{genmo} \times \underline{tm}$	$\rightarrow \underline{genmo}$	Table 9
	get_ref	$\underline{genmo}$	$\rightarrow \underline{set(ioref)}$	Table 9
	contains	$\underline{set(ioref)} \times \underline{int}$	$\rightarrow \underline{bool}$	Sec. 6.3
Q8	subtype	$\underline{busstop}$	$< \underline{genloc}$	Table 10
	at	$\underline{genmo} \times \underline{tm}$	$\rightarrow \underline{genmo}$	Table 9
	at	$\underline{genmo} \times \underline{genloc}$	$\rightarrow \underline{genmo}$	Table 8
	deftime	$\underline{genmo}$	$\rightarrow \underline{periods}$	Table 6
	duration	$\underline{periods}$	$\rightarrow \underline{real}$	Table 6

No.	Operator	Signature		Def.
Q9	**contains**	$set(region) \times region$	$\rightarrow bool$	Sec. 6.3
	freespace	$groom$	$\rightarrow region$	Table 11
	contains	$set(string) \times string$	$\rightarrow bool$	Sec. 6.3
Q10	**freespace**	$mpptn$	$\rightarrow mpoint$	Table 11
	passes	$mpoint \times region$	$\rightarrow bool$	[20]
Q11	**ref_id**	$mpptn$	$\rightarrow int$	Table 9
	distance	$mpoint \times mpoint$	$\rightarrow mreal$	[20]
	freespace	$mpptn$	$\rightarrow mpoint$	Table 11
	freespace	$genmo$	$\rightarrow mpoint$	Table 11
	at	$genmo \times tm$	$\rightarrow genmo$	Table 9
Q12	subtype	$busroute$	$< genrange$	Table 10
	subtype	$line$	$< genrange$	Table 10
	intersects	$genrange \times genrange$	$\rightarrow bool$	Table 7
Q13	**atperiods**	$genmo \times periods$	$\rightarrow genmo$	Table 6
	at	$genmo \times tm$	$\rightarrow genmo$	Table 9
	get_ref	$genmo$	$\rightarrow set(ioref)$	Table 9
	contains	$set(ioref) \times int$	$\rightarrow bool$	Sec. 6.3
Q14	**at**	$genmo \times tm$	$\rightarrow genmo$	Table 9
	trajectory	$genmo$	$\rightarrow genrange$	Table 8
Q15	subtype	$groom$	$< genrange$	Table 10
	at	$genmo \times genrange$	$\rightarrow genmo$	Table 8
	deftime	$genmo$	$\rightarrow periods$	Table 6
	components	$periods$	$\rightarrow set(periods)$	Sec. 6.3
	duration	$periods$	$\rightarrow real$	Table 6
Q16	**ref_id**	$mpptn$	$\rightarrow int$	Table 9
	subtype	$busstop$	$< genloc$	Table 10
	genloc	$int \times real \times real$	$\rightarrow genloc$	Table 11
	at	$genmo \times genloc$	$\rightarrow genmo$	Table 8
	initial	$genmo$	$\rightarrow intime(genloc)$	Table 6
	val	$intime(genloc)$	$\rightarrow genloc$	Table 6
	=	$genloc \times genloc$	$\rightarrow bool$	Table 7
Q17	subtype	$groom$	$< genrange$	Table 10
	passes	$genmo \times genrange$	$\rightarrow bool$	Table 8
	subtype	$busstop$	$< genloc$	Table 10
	atperiods	$genmo \times periods$	$\rightarrow genmo$	Table 6
	at	$genmo \times tm$	$\rightarrow genmo$	Table 9
	initial	$genmo$	$\rightarrow intime(genloc)$	Table 6
	final	$genmo$	$\rightarrow intime(genloc)$	Table 6
	val	$intime(genloc)$	$\rightarrow genloc$	Table 6
	=	$genloc \times genloc$	$\rightarrow bool$	Table 7
Q18	subtype	$gline$	$< genrange$	Table 10
	atperiods	$genmo \times periods$	$\rightarrow genmo$	Table 8
	components	$periods$	$\rightarrow set(periods)$	Sec. 6.3
	final	$genmo$	$\rightarrow intime(genloc)$	Table 6
	val	$intime(genloc)$	$\rightarrow genloc$	Table 6
	at	$genmo \times tm$	$\rightarrow genmo$	Table 9
	inside	$genloc \times genrange$	$\rightarrow bool$	Table 7

Appendix-C: Type System in free space and road network

	$\rightarrow$ BASE	$\underline{int}$, $\underline{real}$, $\underline{string}$, $\underline{bool}$
	$\rightarrow$ TIME	$\underline{instant}$
BASE $\cup$ TIME	$\rightarrow$ RANGE	$\underline{range}$
	$\rightarrow$ SPATIAL	$\underline{point}$, $\underline{points}$, $\underline{line}$, $\underline{region}$
	$\rightarrow$ GRAPH	$\underline{gpoint}$, $\underline{gline}$
SPATIAL $\cup$ GRAPH	$\rightarrow$ TEMPORAL	$\underline{moving}$, $\underline{intime}$

Table 13: Data Types in [14, 20, 18]

Definition 10.9

UBool = $\{(i, b) | i \in D_{\underline{interval}}, b \in D_{\underline{bool}}\}$

$D_{\underline{mbool}}$ = $\{< u_1, u_2, ..., u_n > | n \geq 0, n \in D_{\underline{int}}$, *and* $\forall i \in [1, n], u_i \in$ *UBool* $\}$

Definition 10.10

UPoint = $\{(i, p_1, p_2) | i \in D_{\underline{interval}}, p_1, p_2 \in D_{\underline{point}}\}$

$D_{\underline{mpoint}}$ = $\{< u_1, u_2, ..., u_n > | n \geq 0, n \in D_{\underline{int}}$, *and* $\forall i \in [1, n], u_i \in$ *UPoint*$\}$

Definition 10.11

NLoc = $\{(rid, pos, side) | rid \in D_{\underline{int}}, pos \in D_{\underline{real}}, side \in D_{\underline{bool}}\}$

$D_{\underline{gpoint}}$ = $\{(n\_id, gp) | n\_id \in D_{\underline{int}}, gp \in$ *NLoc* $\}$

Definition 10.12

NReg = $\{(rid, pos_1, pos_2) | rid \in D_{\underline{int}}, pos_1, pos_2 \in D_{\underline{real}}\}$

$D_{\underline{gline}}$ = $\{(n\_id, gl) | n\_id \in D_{\underline{int}}, gl \in$ *NReg* $\}$

Definition 10.13

UGPoint = $\{(i, gp_1, gp_2) | i \in D_{\underline{interval}}, gp_1, gp_2 \in$ *NLoc and*

(i) $gp_1.rid$ = $gp_2.rid$;

(ii) $gp_1.side$ = $gp_2.side$ $\}$

$D_{\underline{mgpoint}}$ = $\{< u_1, u_2, ..., u_n > | n \geq 0, n \in D_{\underline{int}}$, *and* $\forall i \in [1, n], u_i \in$ *UGPoint*$\}$