A Conceptual View on Trajectories ⋆

Stefano Spaccapietra [a,*], Christine Parent [b],
Maria Luisa Damiani [a], Jose Antonio de Macedo [a],
Fabio Porto [a], Christelle Vangenot [a]

[a] *EPFL - Ecole Polytechnique Fédérale, Database Laboratory, Lausanne, Switzerland*

[b] *UNIL-ISI, Lausanne, Switzerland*

Abstract

Analysis of trajectory data is the key to a growing number of applications aiming at global understanding and management of complex phenomena that involve moving objects (e.g. worldwide courier distribution, city traffic management, bird migration monitoring). Current DBMS support for such data is limited to the ability to store and query raw movement (i.e. the spatio-temporal position of an object). This paper explores how conceptual modeling could provide applications with direct support of trajectories (i.e. movement data that is structured into countable semantic units) as a first class concept. A specific concern is to allow enriching trajectories with semantic annotations allowing users to attach semantic data to specific parts of the trajectory. Building on a preliminary requirement analysis and an application example, the paper proposes two modeling approaches, one based on a design pattern, the other based on dedicated data types, and illustrates their differences in terms of implementation in an extended-relational context.

Key words: Trajectories, moving objects, spatio-temporal databases, GIS

⋆ This work is done in the context of GeoPKDD, an EEC project (IST-6FP-014915) supported by the FP6 programme of the European Community. The project homepage is available at http://geopkdd.isti.cnr.it/. Support is also received from SNF, Swiss National Foundation, grant #200021-116647.

* Corresponding author

Submitted to publication *29 May 2007*

1 Introduction

Thanks to current sensors and GPS technologies, large scale capture of the evolving position of individual mobile objects has become technically and economically feasible. This opens new perspectives for a large number of applications (from e.g. transportation and logistics to ecology and anthropology) built on the knowledge of movements of objects. Typical examples of moving objects include cars, persons and planes equipped with a GPS device, animals bearing a transmitter whose signals are captured by satellites, and parcels tagged with RFIDs. Extending the limited capabilities of commercial data management system, some research prototype systems [1] [27] do provide nowadays support for storing and querying the position of a moving object (i.e. an object whose location changes over time) all along the lifespan of the object. Güting's approach [15] also supports moving regions, which allows for example recording the changing geometry of pollution clouds and flooding waters. Generically speaking, the geometry of a moving object can be of any spatial type and is defined by a function from a temporal domain to a range of spatial values. For simplicity, but without loss of generality, we focus hereinafter on moving objects with point geometry (moving points, in short).

While the ability to record continuous movement is the foundation of managing movement, satisfying application requirements requires more than that. Namely many applications need a more structured recording of movement, i.e. as a temporal sequence of journeys, each one occupying a time interval in the object's lifespan and taking the object from a departure point to a destination point. Daily trips of employees going from home to work and back, weekly journeys of trucks delivering goods to customers distributed within a given region, annual migrations of birds in search of longer daylight, are examples where movement of objects is clearly perceived by the monitoring application as countable traveling units. We refer to these countable journeys as trajectories, and denote the involved object as the traveling object. In all the above examples, a traveling object "produces" many trajectories during its lifespan. Applications collect these trajectories and analyze them, for example to derive mobility patterns to be used as input to some decision making process (e.g. collecting urban trajectories to derive useful knowledge for optimizing traffic management), to acquire more knowledge about the traveling objects (e.g. analyzing bird trajectories to better understand their behavior), or to control the proper implementation of transportation logistics (e.g. monitoring worldwide delivery of parcels in a courier company).

From the application viewpoint, knowledge on trajectories usually includes a multiplicity of semantic data complementing the recording of the moving position. For example, for daily trips of employees applications may wish to know which transportation means have been used during the trip and whether

the trip used carpool facilities. For bird migrations an important information is which weather conditions the birds faced during their flight, and where, why and how long birds stopped on their way. Constraints are also part of the modeling game, for example to rule that birds of certain species never fly during the night.

To fill the gap between the moving objects provided by advanced prototypes and the semantic trajectories needed by applications, a novel approach has to be developed to enable applications to state whatever semantics they want to associate to trajectories. In other words, a conceptual model for trajectories is needed. The conceptual model must allow handling simple trajectories (direct travels from origin to destination) as well as complex ones (where the travel semantically consists of separate sub-travel periods), and associating any kind of semantic annotations to trajectories, be it as attributes of the trajectory or via links between the trajectory and any other object in the database (we call these objects application objects). This mandates that trajectories be seen as a first-class concept in the data model, which is beyond the capability of current spatio-temporal prototypes.

Elaborating a conceptual support is not an obvious task. Trajectories seen as an object of interest hold a complex information structure, showing a mix of generic, application independent, features and many application dependent features. The proposed solution therefore needs to include both standard constructs enriching the underlying spatio-temporal data model and customizable constructs with maximum flexibility that can be easily tailored to the specific semantics of trajectories in a given application context.

This paper proposes two alternative approaches for trajectory modeling, one based on data types and one based on design patterns. The former is inspired by previous work we have done to extend a traditional conceptual model to cover space and time modeling dimensions, thus defining a new spatio-temporal conceptual model [24][25]. The latter resumes an old idea that the conceptual modeling community may have somehow neglected but which we use here as it offers the flexibility we are looking for to cope with the potential complexity of trajectory semantics. The two approaches may be used together if the application requirements suggest that both are useful in a specific design task.

The paper is organized as follows. The next section offers an insight into related work. Section 3 outlines an application scenario we use to enrich the presentation with examples. Section 4 discusses definitions for the basic components of a trajectory approach, while Section 5 explores the application requirements related to the description of these components. Section 6 shows how the two approaches we propose take the requirements into account. Section 7 briefly compares the two approaches, sketching their implementation in

an extended-relational context. Finally, Section 8 concludes pointing to further future work.

2 Related Work

Several research communities have a definite interest in analyzing spatio-temporal phenomena. Trajectory data have been used in e.g. social sciences, biology, medicine, geographical information science, data mining. In social science, for example, the study of human activities and movements in space and time has since long been an important research area, addressing topics such as migration, residential mobility, shopping, travel, and commuting behavior [19][30]. Time-geography [31] is a general framework for the description of human activities in space-time at daily or lifetime scale of observation.

In GIS research, properties of paths describing movement, called geospatial lifelines, have been discussed [21] with a special focus on multi-granularity [17], with the intent to provide their formal characterization. Analyses of groups of geospatial lifelines is used to extract knowledge about movement patterns [20]. In the database community, a landmark contribution is the work by Ralph Güting and his group [15][16], who developed both an in-depth and formal theory for moving objects (points and regions) and a full implementation (in particular using a proprietary extensible DBMS [14]) of their proposal. The core result is the rich set of data types they have defined, covering spatial types, temporal types and moving types. This work provides a solid and well-thought foundation to any further work building on moving objects. Güting's work includes a twofold view of spatio-temporal paths. An abstract view where path description is based on curves defined by infinite sets, and a discrete view where a finite representation approximates the trajectory as a polyline. The discrete model presented is a possible implementation of the abstract model. Recently, Güting et al. have extended their work to modeling and manipulating network-constrained movement, i.e. cases where movement follows a specific network such as a railway or a road network [13]. Most works on network-constrained movement describe, on one side, the network with its geometry and, on another side, the moving object with its coordinates. Then, they constrain the location of the object to be within the geometry of the network. An important advance of [13] is describing the position of the object via linear referencing within the routes described in the network, e.g., on interstate I-55 at 10.3 km after Chicago heading south. Therefore, the moving objects are "naturally" inside the network, and spatial constraints are no longer needed.

An alternative approach to moving objects has been pursued by Wolfson et al. [32]. Their concern about efficiency (in terms of minimizing the update load) led to storing motion vectors rather than spatio-temporal positions. This has

also the advantage to allow predicting future movement according to the current motion vector. The current vector remains active till a difference between the prediction and the sampled position is higher than a given threshold. At that point a new vector is computed and stored. The work only considers moving points, not moving regions. The authors also addressed movement constrained by a network (movement of cars on a road network). More on road-constrained movement can be found in [29].

Building on Güting's and Wolfson's approaches, Pelekis et al. [27] implemented a framework, named HERMES, for moving objects. HERMES is a new data cartridge that exploits Oracle's static spatial data types [23] and temporal literal types (provided by the TAU-TLL temporal data cartridge [28]). HERMES uses Güting's sliced representation of moving values to support different kinds of interpolation functions such as linear and arc sub-motions.

Other issues related to management of moving points have been addressed. For example, how movement of two moving elements (points or regions) may be correlated through various distance measures and how movement of one element may be analyzed in terms of its position relative to the other element [22]. System performance concerns generated further work, including several indexing schemes for moving objects [8] [26].

An interesting complement to nowadays traditional moving points is recent work on periodicity of movement. Movement may indeed be repeated regularly, like the movement of planets, trains, or migrating animals. In [4] the authors define a formal model for representing periodic movements that may contain nested repetitions, as, for instance, the movement of an underground train that covers the same trajectory every weekday. Following the data types approach of Güting's earlier work [15], the authors have defined and implemented on SECONDO [14] a set of data types for periodic movements, in particular the *pmpoint* type (periodic moving point).

While developing a rich body of work for managing moving objects, the database research community has shown very little interest in the application viewpoint on moving objects, i.e. providing support for the concept of trajectories at the conceptual level. State of the art conceptual models for spatio-temporal databases do include the moving point construct e.g. [18][24]. But they haven't taken it any further. The only work we are aware of is a 2004 contribution [5] that aims at enriching the semantics of a moving object model. However, what the authors call trajectory is the polyline connecting the sample points that define the discrete representation of movement, i.e. a sequence of spatio-temporal segments, rather than a sequence of semantic steps. Similarly, the proposed properties of trajectories are those derivable from the raw data (e.g. speed, covered area), not the properties that an application may wish to maintain about trajectories (e.g. their underlying goal).

The work we propose in this paper addresses trajectories as movements that correspond to semantically meaningful travels (of humans, animals, objects, and phenomena). The underlying spatio-temporal path is but one component of a trajectory. Other components include the definition of when a trajectory starts, when it ends, and when it pauses. These components are fixed by the application, based on the semantics given to the trajectories. Therefore these components have to be supported by a conceptual model for trajectories.

Like for other spatio-temporal constructs, applications need the capability to define trajectories at multiple spatial and temporal granularities. Few works addressed multiple representations of spatial on moving objects [11][7]. We rely on our own previous work [3][25] to handle multiple representations of trajectories in the same way we did for other data. This aspect will not be discussed in this paper.

3 Application Scenario

This section sketches an application scenario using trajectory data. The scenario observes migrating birds and collects data sent by transmitters or by capturing the birds and making direct measures.

Several research groups [1] collect data on annual migrations of white storks (*Ciconia ciconia*). Like other species, white storks migrate in search of better food availability. Each autumn they leave the north hemisphere (e.g. Europe) and migrate south (e.g. Africa) where there is an all year round food supply. In spring, they migrate back north to breed, because they need the longer days of the northern summer to feed their young. Analyzing bird migrations is the key to improve human knowledge about many open questions on animal behavior: Is migration innate and somehow genetically programmed, how do birds orientate, how do they manage to fly such long distances, do they fly in groups, where and why do they make stopovers during their migration, which environmental factors influence their migration (topography, weather conditions, predators, food availability), which hazards do they encounter, why some birds die during the migration ...

For this application, a number of white storks have been equipped with a tiny transmitter whose signals are captured by satellites. White storks fly only during the day as they use thermal currents, and stop at night for resting and feeding. They can travel several hundreds of kilometers every day depending of weather condition. They migrate in flocks, whose composition may change at stops. To detect flocks (i.e. storks flying together) the spatio-temporal po-

[1] http://www.storchenhof-loburg.info, http://www.fr.ch/mhn/

sitions of the storks are analyzed. However, only a small percentage of storks are equipped with transmitters, hence the flock to which a stork belongs for a given move may be unknown.

In order to analyze the behavior of white storks during their migration, researchers need to record two types of information:

(1) Information about animal trajectories.

While the storks are flying their spatio-temporal position and altitude are recorded at regular intervals. At stops, the following information may be recorded:

- The kind of stop: nightly stop or longer stop (usually a few days or weeks) for resting and eating.
- Whenever observable, the activities of the bird during the stop: feeding, resting...
- If the stork was temporarily caught during this stop, its weight, percentage of fat, body temperature, and global health condition.

During the moves, the following information may be recorded:

- The altitude at which the stork is flying. This information, as well as weather conditions, natural and artificial objects, varies along the trajectory.
- The flock the stork is part of.

(2) Information about the environmental conditions related to a trajectory. These mainly record:

- Weather conditions: wind (direction, strength), temperature (minimal, maximal), pressure, sky condition (sun, rain, clouds, fog, clear sky).
- Natural objects (e.g. mountains, water extents, deserts), and artificial objects, (e.g. electric lines, antennas, tall buildings, wind turbines) that may be obstacles for the birds and influence their flight behavior (particularly the direction). The threat they pose to birds may cause their death.

4 Basic Definitions for Trajectories

Before we articulate the view of trajectories we adopt to fulfill the requirements above, this section provides the basic definitions on which we build our work. The goal in developing a conceptual model for trajectories is to provide constructs and rules that enable the designer of a database that uses movement data to think about these data as sets of identifiable trajectories traveled by application objects. From users' viewpoint, the concept of trajectory is rooted

in the evolving position of some object traveling in some space during a given time interval. Thus, trajectory is by definition a spatio-temporal concept. But while *moving* may be seen as a characteristic of some objects that differentiates them from non-moving objects (e.g. buildings, roads), the concept of traveling object implies that its movement is intended to fulfill a meaningful goal that requires traveling from one place to another. Traveling for achieving a goal takes a finite amount of time (and covers some distance in space), therefore trajectories are inherently defined by a time interval. This time interval is delimited by the instant when the object starts a travel (called t_{begin}) and the instant when the travel terminates (t_{end}) . Identifying t_{begin} and t_{end} within the whole time-frame where the object is moving is an application decision, i.e. a user-driven specification. The following definition formally defines a moving point trajectory in a database perspective.

Definition 1 (Trajectory) *A trajectory is the user defined record of the evolution of the position (perceived as a point) of an object that is moving in space during a given time interval in order to achieve a given goal.*

$$trajectory : [t_{begin}, t_{end}] \rightarrow space$$

This definition settles trajectories as semantic objects. The time space function is defined by the user and is not necessarily the one provided by the data acquisition mechanism. The latter is the raw data, whose form usually is as a sequence of (sample point, time) pairs. Raw data often needs to undergo a cleaning process to correct errors and approximations in data acquisition. In addition, the application may be interested in only a subset of the cleaned sample points, e.g. skipping points acquired during the night to only retain daylight movement or replacing a sequence of irrelevant (from the application perspective) points with a single representative point (e.g. for representing stops as a single point in the sense we define hereinafter).

Another important difference between an object with trajectories and a moving object [15] (also called object with a time-varying geometry in [24]) is that, for a moving object, the time space function that describes its position is defined over the whole lifespan of the object. Instead, a trajectory is given by restricting the function to a specific time interval, $[t_{begin}, t_{end}]$, included in the lifespan of the object. A trajectory is a segment of the spatio-temporal path covered by a moving object (cf. Figure 1). Consequently, during its lifespan an object can travel a number of trajectories, one after the other. For example, a bird has a migration trajectory in spring 2006 from Africa to Europe, followed by another trajectory in autumn 2006 from Europe to Africa, followed by other trajectories in 2007, etc.

As stated, at the conceptual level the segmentation of the path into trajectories is driven by the semantics the application attaches to trajectories. For

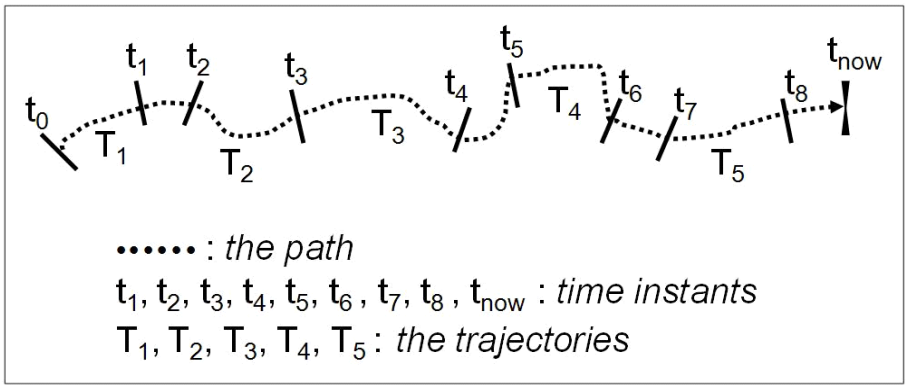

Fig. 1. A spatio-temporal path for an ongoing moving object and its many trajectories defined by a semantic segmentation of the path. In this example parts of the path do not belong to any trajectory: They correspond to movement that is irrelevant to the application.

example, in the bird monitoring application it is up to the ornithologists to define if they see an annual return flight as a single trajectory, or each annual one-way flight as a different trajectory. In some application domains, e.g. goods delivery, trajectories may be easily identified and separated, for example as the daily route of a truck. In other application domains, e.g. when monitoring apes, it is not obvious to determine if the ape starts a new trajectory or just continues the previous one. The segmentation may just be driven by the observation period (e.g. assuming one daily trajectory for each ape, corresponding to the observation period from 8am to 5pm when the observer is at work). However, whatever the application criterion is, each trajectory has a defining time interval $[t_{begin}, t_{end}]$ that is necessarily included in the lifespan of its traveling object and is necessarily disjoint from (or meeting) the time intervals of the other trajectories of the same object .

An open question related to the segmentation of a spatio-temporal path into trajectories is whether the trajectories cover the whole path or not. In the first case, the time intervals of two successive trajectories always meet each other. The answer to this question depends upon the application.

Traveling objects do not necessarily continuously move during a trajectory (this is the case in our example applications). Consequently, trajectories may themselves be semantically segmented by defining a temporal sequence of time sub-intervals where alternatively the object position changes and stays fixed. We call the former the *moves* and the latter the *stops*. We can then see a trajectory as a sequence of moves going from one stop to the next one (or as a sequence of stops separating the moves). For example, a bird that has departed for migration will make a stop somewhere for some time for feeding, another stop for resting, and so on till it reaches the end of its trajectory. Salespersons on a business trip will stop at all locations where they planned meeting a customer.

As already stated, identifying stops (and moves) within a trajectory is the responsibility of the application. Physical stops (i.e. the fact that the position of the object is the same during two or more consecutive instants) do not account for conceptual stops, as they may be due to events that are irrelevant to the application. For example, the stop made by salesperson to drink a coffee is irrelevant for the company's tracking application. Instead, the stop made for meeting a customer is significant. The application may be interested in counting the number of stops per trajectory, and obviously the stops to be counted are only the significant stops. Hereinafter we assume that moves and stops fully cover the trajectory (i.e. there is no instant within $[t_{begin}, t_{end}]$ that belongs neither to a move nor to a stop) .

We semantically define stops and moves as follows.

Definition 2 (Stop) *A stop is a part of a trajectory, such that:*

- *The user has explicitly defined this part of the trajectory ($[t_{beginstopx}, t_{endstopx}]$) to represent a stop.*
- *The temporal extent $[t_{beginstopx}, t_{endstopx}]$ is a non empty time interval, and*
- *The traveling object does not move (as far as the application view of this trajectory is concerned), i.e. the spatial range of the trajectory for the $[t_{beginstopx}, t_{endstopx}]$ interval is a single point. All stops are temporally disjoint, i.e. the temporal extents of two stops are always disjoint.*

Definition 3 (Move) *A move is a part of a trajectory, such that:*

- *The part is delimited by two extremities that represent either two consecutive stops, or t_{begin} and the first stop, or the last stop and t_{end}, or $[t_{begin}, t_{end}]$.*
- *The temporal extent $[t_{beginmovex}, t_{endmovex}]$ is a non empty time interval, and*
- *The spatial range of the trajectory for the $[t_{beginmovex}, t_{endmovex}]$ interval is the spatio-temporal line (not a point) defined by the trajectory function (in fact, it is the polyline built upon the sample points in the $[t_{beginmovex}, t_{endmovex}]$ interval).*

From a database design point of view, a move is a time-varying point defined on the time interval $[t_{beginmovex}, t_{endmovex}]$.

We do not consider begin and end of a trajectory to be stops: Their temporal extent is a single chronon.

5 Requirements for Trajectory Modeling

The previous definitions hold for any kind of trajectory. But modeling requirements may vary according to which kind of trajectories is considered.

We differentiate between three kinds: metaphorical, geographic, and spatio-temporal.

5.1 Metaphorical Trajectories

The term trajectory is sometimes used in a metaphorical sense to describe an evolution, although the evolution at hand is not related to physical movement. For example, it is not uncommon to talk about the career trajectory of a person to describe something like "*I went from academia to industry to a large service corporation and finally back to academia*". This metaphorical use relies on the idea of an object (e.g. the person) moving in an abstract space whose points are the different values of a "*jobSector*" attribute.

From the data modeling perspective, this type of trajectories can be described by defining a time-varying attribute (e.g. *jobSector*) for the traveling object type. Conversely, any time-varying attribute in an object type can be seen as defining a metaphorical trajectory for objects of the type. State-of-the-art data models for spatio-temporal databases support time-varying attributes [18][24]. Variability may be discrete (values exist at some instants only), stepwise (value changes are instantaneous and each value holds for a time interval) and continuous (value changes continuously). Metaphorical trajectories may be of any of these three kinds, depending on the value domain for the time-varying attribute. Continuous variability is obviously only possible if the value domain is continuous. Stepwise and discrete variability apply to enumerated domains and more generically to domains where the semantics of a value change is an update semantics (a new value replaces the current value), not an evolution semantics (a new value results from some "evolution" from the current value). For example, the career trajectory is a stepwise trajectory. Trajectories are discrete if only value change events are relevant, not the state of the value at any point in time. The trajectory of the price for a stock is an example of continuous variation.

As long as the evolution only records the changing values of the evolving attributes, this kind of trajectories does not call for new modeling constructs. However, if the description of a metaphorical trajectory involves links between the trajectory and other application objects, the trajectory has to be modeled as an object, not as an attribute. Consequently, for these semantically richer trajectories the metaphor should be extended to model them alike spatio-temporal trajectories (as discussed below). Obviously, topological and synchronization relationships used to model spatio-temporal trajectories would need to be redefined to apply to metaphorical trajectories.

5.2 Naïve Geographical Trajectories

People most frequently describe their travels as going from place to place, for example from city to city: "*I went to Paris, then to Brussels, then moved to Amsterdam and Berlin before coming back to Lausanne*". Stops here have a strong geographical connotation (hence we qualify these trajectories as geographical), but they are not defined in terms of spatial coordinates (hence the reference to naïve geography [9]). Similarly, moves may be defined in terms of train names, e.g. using TGV from Lausanne to Paris, then Thalys from Paris to Brussels, etc. In fact, geographical trajectories are just a special case of metaphorical trajectories. As the latter, geographical trajectories may be described using time-varying attributes (e.g. *visitedCity*, *boardedTrain*) or in a way similar to spatio-temporal trajectories.

5.3 Spatio-Temporal Trajectories

The original sense of the term trajectory denotes the changing position of an object in geographical space, be it a 3D space (e.g. the trajectory of a plane) or a 2D space (e.g. the trajectory of a rolling ball in a bowling game). We say a trajectory is spatio-temporal if spatial coordinates are used to express the position of the traveling object. Most frequently, the traveling object is geometrically represented as a point (e.g., a person, an animal, a car, a truck, a plane, a ship, a train). Yet the traveling object may have a surface or volume geometry (e.g. clouds, floods, air pollutions, oil spills, avalanches), in which case both change in position and change in shape may concur to define the trajectory. In this paper we only consider modeling spatio-temporal trajectories generated by objects represented as points.

A trajectory has two facets:

- A *geometric facet*: This is the spatio-temporal recording of the position of the traveling point. It is a delimited segment (i.e., a single continuous subset) of the spatio-temporal path covered by the object's position during the whole lifespan of the object. From the conceptual modeling perspective, we can basically rely on Definition 1 and represent the geometric facet as a continuous function from a given time interval into a geographical space (the range of the function): $trajectory : [t_{begin}, t_{end}] \rightarrow space$. However, the modeling structure should also include the sample points (and the interpolation functions) that are used to discretely capture the trajectory function. The geometric facet could be modeled using the moving point data type. We will not do this, as we want to complement the strict geometrical aspect with the definition of stops and moves that are inherent to the semantics of

a trajectory.
- A *semantic facet*: This is the information that conveys the application-oriented meaning of the trajectory and its related characteristics. The semantic facet is analyzed next.

5.4 On Semantics of Trajectories

Which semantic characteristics of trajectories are relevant to the conceptual view in a specific application depends on the application requirements. However, a generic characterization can be outlined. A very basic semantic concern is supporting the definition of the stops, if any, which decompose the trajectory into a sequence of moves. Stop definition may undergo a number of application dependent constraints, e.g. a limited number of stops are allowed, or stops may have to conform to a minimal/maximal duration or distance between stops.

Another standard semantic concern is giving a meaning to trajectory components, i.e. defining their semantic interpretation in terms of the application objects they represent. This holds for the different components:

- *Stops*. For example, if trajectories are seen by the application as moves between cities, the database must be able to store and return in which city a stop is located. The same trajectories may be seen by another application as moves between countries, thus calling for a link between stops and countries. For migrating birds, stops may be in geographical regions of interest to the birds or to the ornithologist.
- *Moves*. For example, an application monitoring people' use of a train network may need to record which train has been used for which move, e.g. a trajectory of a traveling person may consist of a first move using train 324 on March 13, 2007 from Lausanne to Geneva and a second move using train 278 on March 13, 2007 from Geneva to Lyon.
- *Begin* and *end* of the trajectory. For example, a company that monitors business trips done by its salespersons to meet company's customers may restrict trajectories to originate and end in the company's premises.

Very frequently human trajectories are constrained to follow a specific network, e.g. cars and trucks can only move on roads, traveling by train/bus can only use trajectories consistent with the underlying train/bus network, planes normally fly within specific corridors, and buses can only stop at bus stops. Network constrained trajectories are characterized by the fact that the whole trajectory geometry has to conform to the geometry of the supporting network, i.e. obey topological inclusion constraints. A detailed analysis of how to describe a road network and trajectories inside the network has been developed by Güting et al. [13].

Applications may wish to maintain various characteristics related to movement, e.g. direction, speed, acceleration (instantaneous or average). Information of this kind is computable from the spatio-temporal function that defines the trajectory, and is usually provided by methods attached to the moving data types.

Finally, trajectory characterization may include any number of semantic properties and integrity constraints (on trajectories as a whole and on its components) that are useful for the application at hand. Defining properties on trajectories is usually referred to as a semantic annotation process. Examples include the type of activity of the traveling object during moves and/or stops, the name of the current location, the name of the street or road on which a move is performed, the persons met during a stop, the altitude of the bird and the weather during the moves in birds' migrations. Spatial and thematic integrity constraints may also be attached to trajectories. For instance, a car trajectory is always inside the road network that is described in the database, a plane cannot be in flight if there is not a team of pilots and stewards associated to it, storks do not fly during the night, salespersons stops have to include at least one meeting with a customer.

Annotation properties may hold the same value for the whole trajectory (e.g. the goal of the trajectory, the total duration of the moves) or a value for each component they characterize (e.g. average speed per move, activity during each stop). Moreover, their value may be time-varying over the time interval of the trajectory or over the time interval of the moves or stops (e.g. the varying altitude of the stork during the moves). Whenever several time-varying annotations are specified for a trajectory and rely on monitored data acquisition, the application has to specify whether the same sample points are used to capture values for all annotations, or each annotation has its own sampling rules.

Last but not least, trajectory applications (alike all applications dealing with geographic data) have to have the possibility to specify multiple representations of all the above data according to different views, resolutions, and targeted uses.

In summary, a conceptual model for trajectories must support the characterization of trajectories and their components with attributes, semantic constraints, topological constraints, and links to application objects. All trajectory components are spatio-temporal data, with begin, end and stops having a point geometry, while moves have a time-varying point geometry. From the temporal perspective, begin and end are of type *Instant*, while stops and moves (as trajectories) are of type *TimeInterval* and can thus be annotated with non-varying and varying properties.

6 Conceptual Modeling for Trajectories

We now investigate solutions to support the application requirements we have identified. As mentioned in the previous section, some of the requirements are already covered when using a state-of-art conceptual spatio-temporal data model, of which MADS [24] is a good example. MADS supports spatial and temporal objects and relationships (i.e. objects and relationships that have a geometry attribute describing their spatial extent and have a lifecycle attribute describing their temporal extent (the lifespan), and their activity status, active-suspended-disabled), spatial attributes (whose value domain is a spatial data type), time-varying attributes (recording their values over time), topological and synchronization relationships (spatially and temporally constraining the linked objects), derived attributes, and allows multi-representation at both the schema level and at the instance level. Relying on MADS minimizes the effort needed to extend the data model for support of trajectories.

We have devised two solutions that are driven by different modeling goals. The first solution aims at explicit representation of trajectories and their components (stops, moves, begin and end) in the database schema. Because the requirements call for the possibility to link any of these components to application objects, and because usual conceptual models only support links between object types (not between relationship types and not between an attribute and something else), trajectories and their components are necessarily described by object types. The advantage of this solution is improved readability and understandability of the schema, easiness for the designer to add the semantic information specific to the application, and higher evolvability in terms of future modifications of the initial schema. In this solution the model supports the designer by providing a design pattern, i.e. a predefined sub-schema that provides the basic data structures for trajectory modeling.

The modeling goal that led us to define the alternative solution is to hide as much trajectory data as possible into a dedicated *TrajectoryType* data type, equipped with methods providing access to trajectory components (stops, moves, begin and end). The definition of dedicated data types is a well-known technique to extend a data model to take into account new types of data. It is the technique we have used for example to embed support of spatial and temporal features into MADS. The case for trajectories, however, cannot be fully solved with data types only, as much of the information on trajectories is application-dependent. In this solution, the data type handles the geometric facet of the trajectory and the definition of its components (stops, moves, begin and end). The semantic information that complements trajectories for a given application (attributes describing the trajectory, begin, end, stops, moves, or parts of them, and the relationships linking the trajectory, begin,

end, stops, moves or parts of them) is described as attributes and relationships of the traveling object.

The following sub-sections describe in detail the two solutions.

6.1 Trajectory Design Pattern

The use of design patterns is a well-known technique to simplify the task of the schema designer by providing predefined half-baked schemas to handle recurring situations [12]. A design pattern is a predefined generic schema that may be imported into the schema of the application database and then connected to the rest of the database schema by the designer, who has also the ability to modify and adjust the pattern to the requirements specific to the application at hand. We qualify the pattern as a half-baked schema because of the need to adjust it and connect it to the rest of the schema.

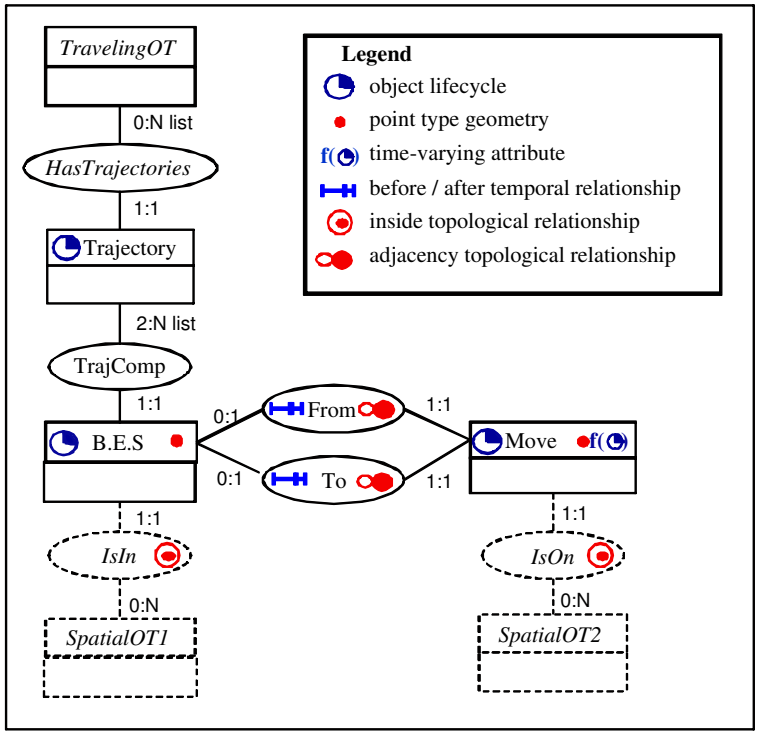

Fig. 2. The proposed pattern for trajectories.

As stated above, a trajectory design pattern holds object types for representing trajectories and their begin, end, stops and moves. In the pattern we propose (one of many possible variants), illustrated in Figure 2, an object type *B.E.S* groups begin, end, and stop objects as instances of the same type, because of their similar features. Each *B.E.S* object has a lifecycle, which is a simple time interval, and a geometry, which is a point. A relationship type *TrajComp* relates trajectories to their components. Its cardinalities enforce that each

component belongs to a single trajectory, and each trajectory has at least two components (begin and end). The object type *Move* has a lifecycle, which is a simple time interval, and a geometry, which is a time-varying point. The two object types *B.E.S* and *Move* are related by two relationship types, *From* and *To*, materializing the fact that each move starts and ends in a stop. Both *From* and *To* bear a topological (adjacency) and a synchronization (meet) constraint enforcing that each move is linked to stops that are adjacent to it in both space and time . The lifecycle of *Trajectory* objects is a time interval which is inferred from the lifecycles of the first and last instances of *B.E.S* to which they are linked (i.e. the instants of its begin and end).

In addition to expressing the internal structure of a trajectory, the pattern includes the hooks used for its connection to application objects. In Figure 2, the names of the hooks are written in italics. Trajectories are linked to a hook object type *TravelingOT* that represents the traveling objects covering the trajectories. Begin, end and stops (*B.E.S*) may be linked to a hook spatial object type *SpatialOT1* that represents the corresponding location in terms of application objects. The *IsIn* relationship bears a topological inside constraint. As this hook is optional, it is drawn with dotted lines in Figure 2. Similarly, moves may be related to a hook object type *SpatialOT2* by another topological inside relationship, called *IsOn*, that may be used to model network-constrained trajectories.

When the pattern is imported, the designers have to adapt it to their application. They may delete the elements the application is not interested in, add more elements to fulfill additional semantic requirements, and modify the pattern structure to make it fully compliant to the requirements. For instance, they may add Begin, End, and Stop object types as subtypes of the *B.E.S* object type. They have to replace by application object types the hook object types that have been kept.

As an example, Figure 3 shows a possible use of the trajectory pattern for the schema of the stork application. Here, personalization of the pattern includes matching the *TravelingOT* object type with the *WhiteStork* object type, matching the *SpatialOT1* object type with the *Country* object type. In addition to matching a hook, the designer deleted the *IsOn* relationship type and added application-specific links to *B.E.S* (to record links to bird zones in addition to links to countries), and *Move* (to record proximity of moves to natural and artificial hazards and record which moves the bird made within which flocks).

Finally, application attributes have been defined for the pattern object types. For example, the direction (*North/South*) and the year of the trajectory, the weather during the trajectory have been added to Trajectory, altitude is provided all along moves, and stops carry information on the kind of stop, activ-

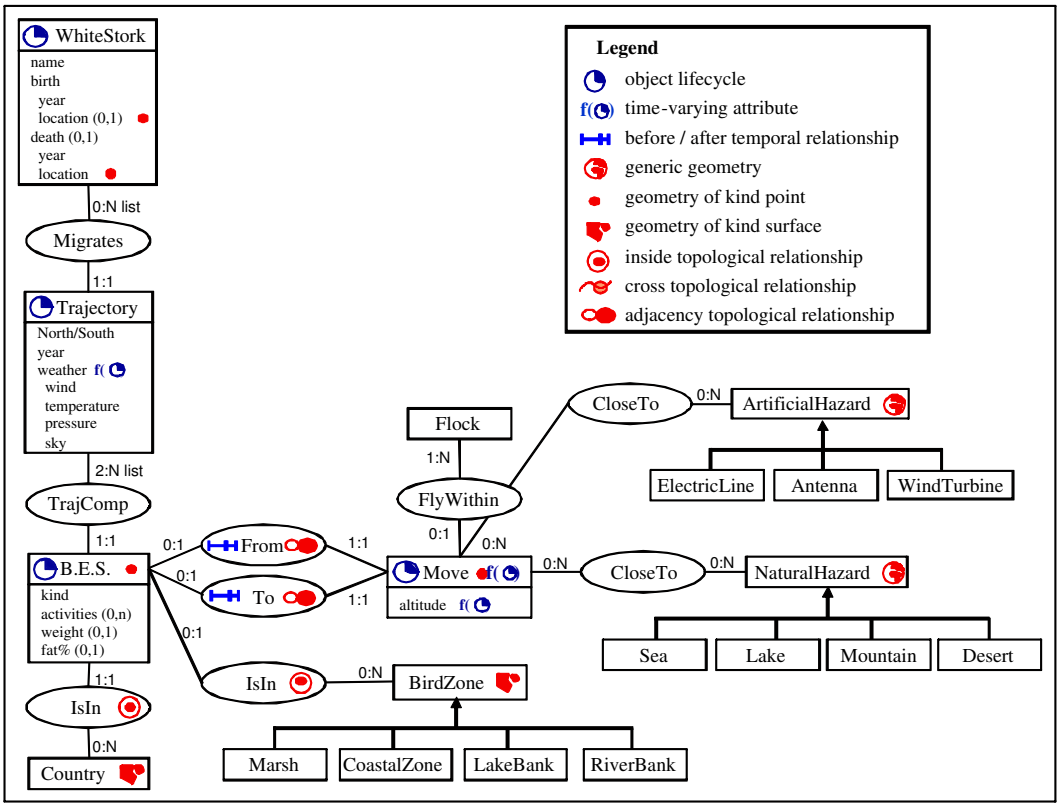

Fig. 3. The pattern personalized for the Stork schema.

ities of the bird, its weight and $fat\%$ at the stop [2] . Both attributes, *weather* and *altitude*, are time-varying, i.e. the history of the weather is recorded during the whole trajectory [3] and the varying altitude of the bird is recorded during each move. Practically, the function specifying a time-varying attribute is defined by a list of couples (value, instant), called sample values (or sample points in the case of a time-varying point) and interpolation functions between the sample points. When an object bears several time-varying attributes, a priori their functions are independent, i.e. their sample values are defined for different instants. If the application requires several attributes to be defined on the same list of instants, the designers have to explicitly state it through an integrity constraint or by grouping the attributes into a complex time-varying attribute. For instance, in Figure 3, *weather* is a complex

[2] It is a design choice to attach the attributes activities, weight and $fat\%$ to the object type whose lifespan corresponds to their period of validity. An alternative choice is to attach these attributes to the traveling object type (*WhiteStork*). In this last solution, the attributes are time-varying in order to define for which stop or move each value is valid. This solution is the one developed for the schema of Figure 4.

[3] At each instant of the lifespan of the trajectory, the weather is recorded at a unique point, the point where the stork is at this instant. Therefore, weather is a time-varying attribute, but not a space and time-varying attribute.

time-varying attribute, whose sample values have the following format:

$$(instant, wind, temperature, pressure, sky).$$

On the other hand, the *Move* object type has two time-varying attributes, *geometry* [4] and *altitude* that could not be grouped into a complex attribute because *geometry* is a system defined attribute. An integrity constraint is needed to express: "*The lists of sample values of* geometry *and* altitude *share the same list of instants*".

Lastly, in order to provide users with an easy access to information, the designers chose to record some redundancy, that they explicited through integrity constraints. For instance, the *birth.year* attribute of *WhiteStork* is defined as derived from the beginning instant of the object's lifecycle, and the *North/South* attribute of *Trajectory* as derived from the spatial positions of the beginning and end of the trajectory (i.e. the positions of the first and last linked *B.E.S* objects).

6.2 Trajectory Data Types

Addressing trajectory-modeling requirements through the use of dedicated data types is the second solution we propose. Following on the discussion above, a generic model of trajectories shall include concepts for describing their begin, end, moves, stops, as well as their sample points and interpolation functions. This is the minimal information that is common to all trajectories, whatever their semantics, and consequently it is the information that can be encapsulated into a generic data type. More semantic information on trajectories is expected to be required, but this information is application specific and cannot be encapsulated into the data type. It has to be explicitly defined by the database designer. Consequently, a data type approach has to be articulated into two complementary facets:

- Relying on generic data types to hold the geometric component of trajectories (i.e. the definition of the sample points and the corresponding interpolation functions) and the definition of the begin, end, stops, and moves.
- Explicitly modeling the application-dependent features through attributes of the object type that represents the traveling object or its trajectories, and relationships linking this object type to application objects.

This twofold approach materializes as follow. We first precisely define the data types.

[4] In the MADS data model, geometry is the system defined name of the attribute that contains the spatial extent of a spatial object (or relationship) type.

6.2.1 Data type TrajectoryType

Each element of *TrajectoryType* describes a single trajectory and consists of:

- A time-varying point representing the spatio-temporal trajectory;
- A list of sample points (of type *Tuple(Point, Instant)*) that implement the function of the time-varying point ;
- A list of stops, such that each stop is a restriction of the function of the time-varying point to a true (i.e. not empty) interval and its spatial range is a point. All time intervals associated with stops must be disjoint;
- A list of moves, such that each move is a restriction of the function of the time-varying point to the time interval that lies between two consecutive stops (or between the beginning of the trajectory and the first stop, or between the last stop and the end of the trajectory).

As *TrajectoryType* is a rather complex type, the definition of its set of methods relies on several choices. Methods returning an element at a time are more useful when writing programs, while methods returning a set of elements are more useful when writing SQL queries. Moreover, a method that returns a set of elements with associated valid times may present its result either as a mere table (element, valid time) or as a time-varying type. With the latter presentation, users may benefit from all the methods of the varying data type. In the sequel, we present a few examples of these different kinds of methods. First, two data acquisition methods:

- `defineSamplePoints(s Table(p Point, t Instant))` - to define the set `s` of sample points of the trajectory (including the begin and end points). `s` is a relational table. Precondition: There is no couple of tuples with the same `t` value.
- `defineStops(s Table(t1 Instant, t2 Instant)` - to define the list of stops of the trajectory - and consequently the list of moves. `s` is a relational table. Each tuple of `s` specifies the restriction of the time-varying function to the time interval $[t1, t2]$. Precondition: For each tuple, the attributes t1 and t2 must define a non-empty time interval. All time intervals $[t1, t2]$ must be disjoint. The spatial extent of each restriction of the function of the time-varying point to $[t1, t2]$ must be a single point (non moving).

Examples of data retrieval methods follow, showing two versions of the same functionality, once with a time-varying result and once with a table result.

- `begin()` $\rightarrow$ `Tuple(p Point, t Instant)` - returns the begin of the trajectory.
- `end()` $\rightarrow$ `Tuple(p Point, t Instant)` - returns the end of the trajectory.
- `samplePointsVarying()` $\rightarrow$ `Varying(Time, Point)` - returns the list of sample points as a discrete time-varying point.
- `samplePointsTable()` $\rightarrow$ `Table(p Point, t Instant)` - returns the list

of sample points as a relational table.
- `stopsVarying()` → `Varying(Time, Point)` - returns the list of stops as a stepwise time-varying Tuple(n Integer, p Point) . The attribute `n` contains the sequence number of the stop.
- `stopsTable()` → `Table(n Integer, p Point, t1 Instant, t2 Instant)` - returns the list of stops as a relational table. The attribute `n` contains the sequence number of the stop.
- `stopAt(t Instant)` → `Tuple(n Integer, p Point, t1 Instant, t2 Instant)` - returns the stop that is ongoing at the `t` instant (if it exists).
- `stop(n Integer)` → `Tuple(p Point, t1 Instant, t2 Instant)` - returns the n^{th} stop (if it exists).

Similar methods may be defined for moves. Lastly, a few examples of computational methods retrieving derived data follow.

- `averageSpeed()` → `Real` - returns the average speed of the traveling object along the whole trajectory.
- `numberOfStops()` → `Integer` - returns the number of stops (excluding begin and end) in the trajectory.
- `orientation(n Integer)` → `Real` - returns the average orientation (in degrees) of the n^{th} move. It is the orientation of the segment defined by the spatial positions of the $n-1^{th}$ and n^{th} stops.

When the result of a method is a time-varying point, users may then use the methods of this latter data type. For instance, if a user wants to know the duration of the trajectory of an object, `o`, (s)he can write:

```
o.samplePointsVarying().defTime().duration()
```

where `defTime()` is the method that returns the temporal extent of a time varying element, and `duration()` is a method that computes the duration of any temporal element. If a user wants the collection of points corresponding to the stops (the spatial locations of the stops) of the trajectory of o, (s)he can write: `o.stopsVarying().rangeValues()` where `rangeValues()` is the method that returns the range of a time varying element. This expression returns the bag of points corresponding to the stops.

6.2.2 Data type TrajectoryListType

Each element of *TrajectoryListType* describes a list of trajectories (ordered by time) and consists of a list of elements of type *TrajectoryType*, such that their temporal domains are disjoint or adjacent. The methods of *TrajectoryListType* are those of the generic List data type. Alternatively, methods of the LIST data type can be reformulated for trajectories, as in the following example:

`trajectory(n Integer)` → ***`TrajectoryType`*** - returns the n^{th} trajectory of the list.

Two examples of methods specific to the *TrajectoryListType* are:

- `trajectoryAtInstant(t Instant)` → ***`TrajectoryType`*** - returns the trajectory that exists at instant `t`.
- `durationBetween(n Integer)` → `Real` - returns the time duration between the n^{th} and the $n + 1^{th}$ trajectories.

6.2.3 Relationships linking a traveling object type and its trajectories

For applications that only require geometric data on trajectories, defining the traveling object type as having a geometry of type *TrajectoryType* or *TrajectoryTypeList* is all what is needed. In the other cases, the semantic information on the trajectory has to be added by the designer either as properties of the traveling or trajectory object types or via relationships on these object types.

For example, the white stork application needs to record many trajectories for each bird, and to know for each trajectory its direction and weather data, as well as the *activity*, *altitude*, *weight* and *fat%* of the bird during the travel. For readability of the schema, the designers have chosen to define two object types, *WhiteStork* and *Trajectory*, linked by the *Migrates* relationship type (cf. Figure 4). Instances of *Trajectory* represent single travels. Information directly related to a trajectory is then defined as attributes of Trajectory and Trajectory is a spatial object type with a geometry of type *TrajectoryType*. Attributes whose value changes during the trajectory are defined as time-varying. Their temporal domains are either equal to the whole lifespan of the trajectory or parts of it, depending on whether they are valued during the whole trajectory or during some parts only. For instance, the time-varying *weather* attribute is defined for the whole lifespan of the trajectory. On the other hand *altitude* is defined during the moves only, while *activity* and *health* are defined during stops only. Therefore, *altitude* is defined on the set of time intervals that correspond to the *moves*, *activity* and *health* on the set of time intervals that correspond to the stops. The health attribute groups two component attributes, *weight* and *fat%*, whose values are measured simultaneously, each time the bird is caught at a stop.

The semantic interpretation of trajectory components (i.e. what do begin, end, stops and moves represent) is available via the *IsIN* relationship types. As in the pattern-based schema, these relationship types bear a topological inside constraint. As trajectories have a time-varying geometry, the topological predicate is satisfied if and only if it is true at any instant of the temporal

extent of the trajectory [5]. If the relationship *IsIN* linking *Country* were not temporal, its topological constraint would read:

> An instance, s, of *Trajectory* may be linked by *IsIn* to an instance, c, of *Country* only if at each instant of the s trajectory, the location of s is inside the geometry of c; i.e. the s trajectory is always inside c.

Using this construction is appropriate if trajectories travel over a single country. In reality, stork migrations traverse several countries. Therefore the schema for our application, shown Figure 4, defines *IsIn* as a relationship type with lifecycle (in addition to the topological constraint). This entails that an instance of *IsIn* has a limited lifespan which is defined as denoting the temporal interval the stork has been flying over (or stopping in) the linked country. A trajectory may then be linked by several *IsIn* instances to different countries at different times. The topological constraint only applies when the relationship instance is active. It now reads:

> An instance, s, of *Trajectory* may be linked by *IsIn* to an instance, c, of Country only if at each instant belonging to the intersection of the temporal extent of the s trajectory and the activespan of the *IsIn* instance, the location of s is inside the geometry of c; i.e. during the lifespan of the relationship the trajectory is always inside c.

The same solution, a relationship type with lifecycle, has been used to state when the trajectory passed close to a natural or artificial hazard and, if known, with which flock the stork was flying during the move. In general, attaching lifecycle information to a link between the trajectory and application objects (in this example to *IsIn*, *CloseTo*, *BelongsTo*) allows recording the fact that a trajectory is successively linked to different objects of the same type for different periods of time. This is represented by creating as many instances of the relationship type, each instance having a different validity lifespan. The goal is to define a sort of time-varying role in the link between one instance on one side (e.g. a given trajectory) and a collection of instances on the other side (e.g. the list of countries the trajectory traverses), with the lifecycle expressing which is the linked application object for each time interval. Obviously, the lifespan in the relationship instances is constrained to be within the temporal extent of the trajectory. And, as for attributes, if the link holds only for some parts of the trajectory (e.g. only for stops or only for moves) the lifespan has to be correspondingly restricted. For instance, the lifecycle of *BelongsTo* is equal to the set of time intervals of the moves. Moreover, if the semantics of the link to objects of some type is that a trajectory has to be linked to only one object

[5] There are in fact two variants of this topological predicate for time-varying geometries. The one used here is the variant that requires the predicate to be true for all instants in the considered time interval. The other variant defines the predicate as satisfied if it is true for at least one instant in the considered time interval.

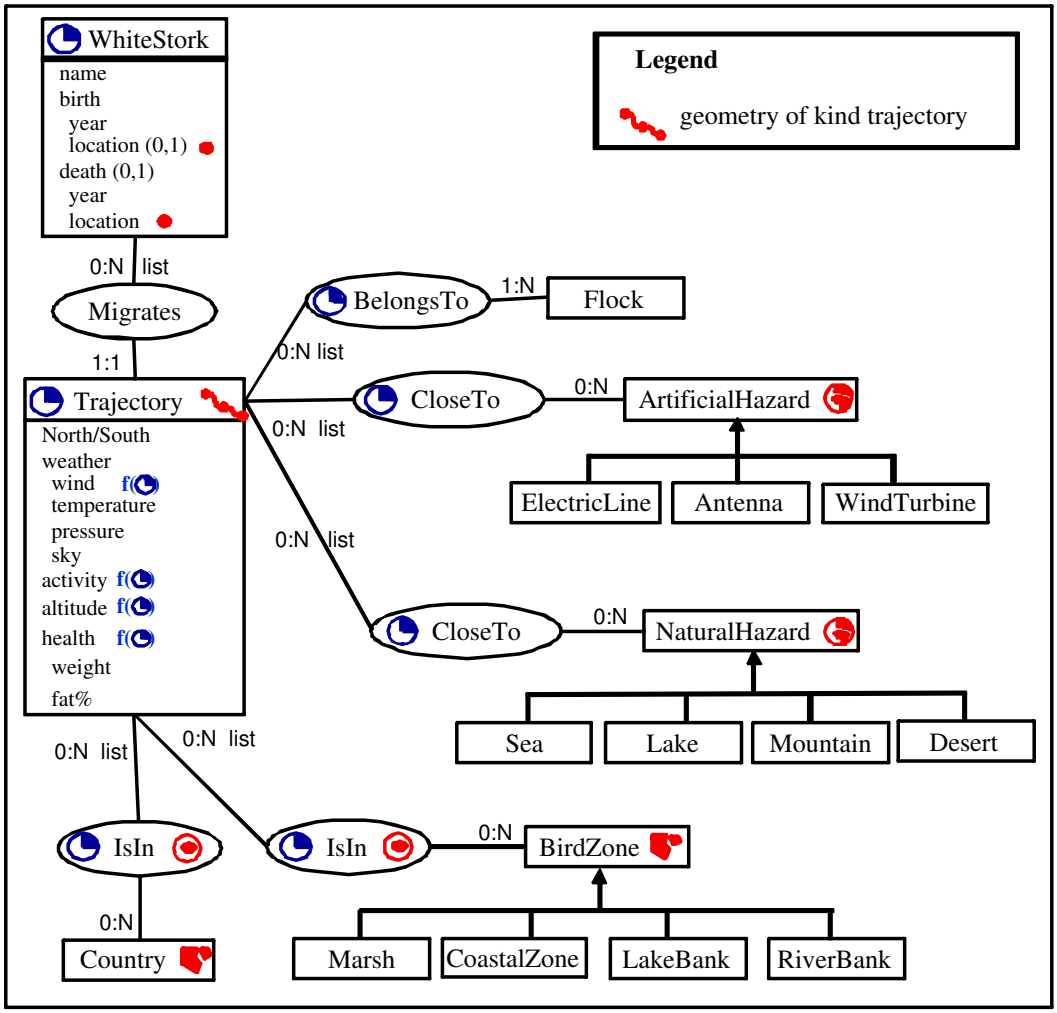

Fig. 4. A data type-based schema for the *White Stork* database.

at a time, instant cardinalities may be specified. For example, during a move a stork belongs to exactly one flock (which may be unknown). Therefore, the temporal cardinality that counts all present and past instances of relationship instances is, for the role *Trajectory-BelongsTo*, equal to (0,N): A stork may fly during a move with some flock and the next one with another flock. But the instant cardinality is (0,1) as at each instant a flying stork belongs to at most a flock. In order not to overload the figures, only temporal cardinalities are shown.

7 Evaluation of the Approaches

We are currently collecting descriptions of various applications and their trajectory data in order to experiment designing the corresponding databases with the two approaches we have outlined. The goal is to have users assess

their preferences through comparison of the two versions of the schema. Our guess is that the data type approach is preferable when dealing with trajectories without stop or trajectories that only need basic semantics, such as stops are in some kind of surface object and moves are on some kind of network, and not much else. A correct assessment should also cover data manipulation aspects. From this viewpoint, a few observations can be readily stated.

A data type supports a set of methods (see examples in previous section) that help users manipulating the elements of the data type. This kind of manipulation interface can be very powerful, in particular if application-specific methods are added to the basic methods of the data types.

The design pattern-based approach requires a specific training for the database designer, but once the schema is designed, it looks like a normal spatio-temporal schema. Users do not need to be trained in the use of new data types for trajectories and their methods. Their queries are standard SQL-like queries for spatio-temporal databases. While this is a clear advantage, its drawback is that the functionality associated to methods in the data type approach has to be coded either by the application developers or by the users writing more complex queries.

Let us illustrate the difference in manipulation expressions. We assume an implementation of both White Storks schemas on an extended relational DBMS equipped with a set of spatio-temporal data types, including time-varying data types. Fig. 5 shows an excerpt of the relational schema corresponding to the conceptual design-pattern based schema of Fig. 3.

In this schema, the domains of values of the attributes *wind*, *temperature*, *pressure*, *sky*, and *altitude* are time-varying domains, e.g. *wind* is of type time-varying Real. The domain of the attribute *PtLocB* of *BES* is *Point*, and the one of *PtLocM* of *Move* is *Time-Varying Point*. In order to enforce the spatial semantics of the relation *CloseToAH*, each time a tuple is inserted in *CloseToAH*, a trigger checks if the time-varying point, *PtLocM*, of the referenced move effectively passes close to (i.e. at a distance smaller than a fixed threshold) the geometry of the referenced artificial hazard during this move. In the same way, triggers enforce the synchronization and topological constraints of the moves with respect to the stops. For instance a trigger checks that for each tuple of *Move*, *tBeginM* is the next instant after the *tEndB* of the tuple of *BES* referenced by *FromBES*. Another trigger checks that the time-varying point *PtLocM* of *Move* effectively starts at the point, *PtLocB*, of the *BES* tuple referenced by *FromBES*.

The query "*How many times did the white stork Max stop during each of her trajectories?*" can be expressed as follows:

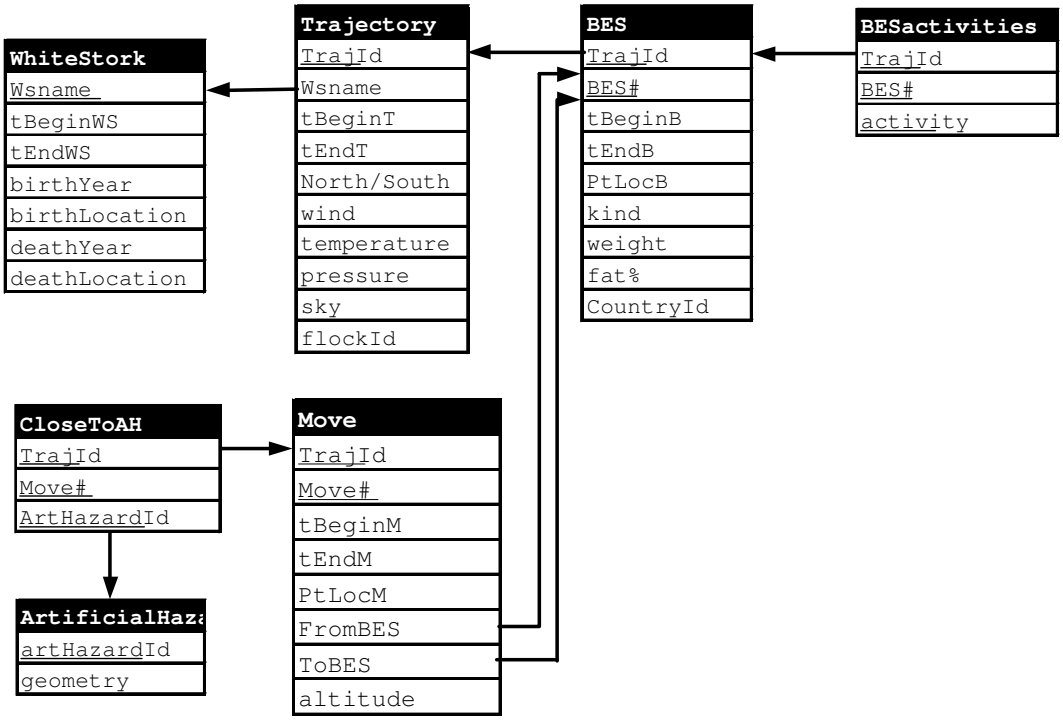

Fig. 5. Excerpt of the relational schema corresponding to Figure 3.

```
SELECT t.TrajId, t.tBeginT, count(b.BES#)
FROM Trajectory AS t, BES AS b
WHERE t.TrajId=b.TrajId AND t.Wsname="Max"
GROUP BY t.trajId, t.tBeginT
```

On the other hand, Fig. 6 partially shows the relational schema that implements the conceptual data type-based schema of Fig. 4.

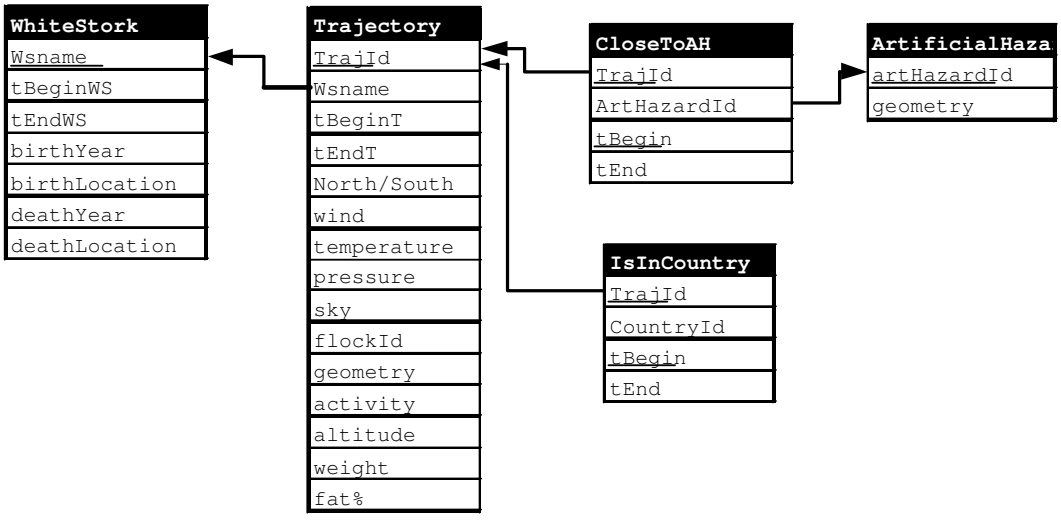

Fig. 6. Excerpt of the relational schema corresponding to Figure 4.

In this schema, the domain of values of geometry of Trajectory is the data type *TrajectoryType*. The domains of the attributes *wind, temperature, pres-*

sure, sky, activity, altitude, weight, and fat% are time-varying domains. The topological and synchronization constraints between stops and moves that had to be specified through triggers in the previous schema are here no longer needed. They are encapsulated in the *TrajectoryType* data type. As in the previous schema, constraints should be enforced for each redundancy. For instance, a CHECK clause in *Trajectory* could enforce that, for each tuple, the beginning (resp. end) of the lifecycle, *tBeginT* (resp. *tEndT*), is equal to the beginning (resp. last) instant of the geometry, `geometry.begin().t` (resp. `geometry.end().t`). On this schema, the same query as above "*How many times did the white stork Max stop during each of her trajectories?*" can be expressed as follows:

```
SELECT TrajId, tBeginT, count(geometry.stopsTable())
FROM Trajectory
WHERE Wsname="Max"
```

In this query, *geometry* is a spatio-temporal trajectory and stopsTable() is the method that returns the set of stops of the trajectory. The expression of the query on the second schema is much simpler: no join, no group by. But users have to know the set of methods associated to the *TrajectoryType* data type.

Let us turn to another query: " *Which activities did the stork have during each stop of trajectory 133?*" On the schema of Fig. 5 (corresponding to the pattern approach), the query is very simple:

```
SELECT BES#, activity
FROM BESactivities
WHERE TrajId=133
```

While on the schema of Fig. 6, it can be written as follows:

```
SELECT s.n, a.value
FROM Trajectory AS t, t.geometry.stopsTable()AS s,
t.activity.sampleValuesTable() AS a
WHERE t.TrajId=133 AND a.tBegin>=s.t1 AND a.tEnd<=s.t2
```

This last query accesses the table containing the stops and the one containing the sample values of the stepwise time-varying *activity* attribute. This latter table is obtained by the method `sampleValuesTable()` of the time-varying data type. This method returns the set of sample values of as a table of format `(value String, tBegin Instant, tEnd Instant)`. This example shows that the *TrajectoryType* solution forces users to express temporal joins when stops (or moves) are accessed with some of their varying attributes. These temporal joins are needed to associate each stop (or move) with the right subset of the attribute values. On the other hand in the pattern solution, attributes that are varying during stops (resp. moves) can be stored as attributes of stops (resp. moves), and temporal joins are not needed.

8 Conclusion

From the application viewpoint, spatio-temporal trajectories of moving objects are complex artifacts that combine raw movement features (where and when the object is) with a variety of semantic annotations that capture the specific knowledge required by each application. These semantic annotations support a meaningful interpretation of the trajectory as a goal-oriented journey of the traveling object. We have sketched two example applications, one on migrating birds, whose goal is search for food, and the other one on business travels of company employees, whose goal is to do business with distant customers.

The contribution of this paper has been: 1) to analyze and emphasize the requirements (from the application viewpoint and in terms of data modeling constructs) for a semantically rich representation of objects' travels, and 2) to consequently propose two conceptual modeling approaches for direct support of trajectory semantics. Eventually, an implementation of the approaches is shown. The work reported is, to the best of our knowledge, the first one to propose a fully conceptual approach for modeling the semantics of moving objects, adding a semantic layer to the usual modeling of trajectories as raw movement of moving objects.

Work is still needed to further explore the interaction between our trajectory modeling strategies and the multiple network models that have been proposed in the literature, in view of enriching, if applicable, the modeling of network-constrained trajectories.

Most of future work will be devoted to similarly defining a modeling strategy for trajectory warehousing. Building a trajectory data warehouse is the preliminary step towards data mining (the core focus of the GeoPKDD project we are involved in). In this context we are highly interested in investigating the interplay between trajectory conceptual modeling and spatio-temporal data mining. Data mining techniques are instrumental in extracting spatio-temporal and semantic patterns [10][2]. These patterns represent abstract trajectories, i.e. they express stereotypes rather than actual trajectories. Conceptual representation of extracted patterns may therefore call for an adjustment of our approach.

A related line of investigation concerns the specification of operators for the aggregation of trajectories. The aggregation concept generically denotes any process somehow collecting a set of data items to produce some global data item that materializes a synthetic view of the collected data items. Different kinds of aggregation are possible and they can be performed at different levels, distinguishing for example among operators applied to the components of a trajectory and operators applied to trajectories as a unit. An example

operation on components is an aggregation replacing a set of stops in a region with a unique stop accounting for the whole time spent in the region. An example operation on trajectories as a unit is an aggregation replacing a set of trajectories with the average trajectory (somehow) computed from the set. Exploring aggregation functionality [6] is a well-known research direction in spatio-temporal warehousing and we intend to extend current techniques to cope with our new concept of semantically enriched trajectories.

Acknowledgements

The authors are definitely indebted to Monica Wachowicz for motivating our interest in the semantics of trajectories with a first informal document for discussion among the GeoPKDD partners. We also extend our thanks to all GeoPKDD partners for their comments on a preliminary presentation of this work at a project meeting.

References

[1] V. T. Almeida, R. H. Güting, and T. Behr. Querying Moving Objects in SECONDO. In *7th International Conference on Mobile Data Management (MDM 2006)*, pages 47–51, Nara, Japan, 2006.

[2] L. Alvares, V. Bogorny, J. Macedo, and B. Moelans. Dynamic modeling of trajectory patterns using data mining and reverse engineering. In *submitted for publication*, 2007.

[3] S. Balley, C. Parent, and S. Spaccapietra. Modeling geographic data with multiple representations. In *International Journal on GIS (IJGIS)*, volume V.18,, pages 329–354. June 2004.

[4] T. Behr, V. T. Almeida, and R. H. Güting. Representation of periodic moving objects in databases. In *14th ACM International Symposium on Geographic Information Systems (ACM-GIS'06)*, pages 43–50, 2006.

[5] S. Brakatsoulas, D. Pfoser, and N. Tryfona. Modeling, storing and mining moving objects databases. In *International Database Engineering and Applications Symposium (IDEAS'04)*, 2004.

[6] F. Braz, S. Orlando, R. Orsini, A. Raffaeta, A. Roncato, and C. Silvestri. Approximate aggregations in trajectory data warehouses. In *Workshop on Spatio-temporal Data Mining (STDM'07) in conjunction with ICDE'07*, Istanbul, April 20 2007.

[7] E. Camossi, M. Bertolotto, E. Bertino, and G. Guerrini. A multigranular spatiotemporal data model. In *Proc. of 11th International Symposium on Advances in Geographic Information Systems (ACM GIS 2003)*, pages 94–101, New Orleans, Louisiana, November 7-8 2003.

[8] V. P. Chakka, A. Everspaugh, and J. M. Patel. Indexing Large Trajectory Data Sets With SETI. In *First Biennial Conference on Innovative Data Systems Research (CIDR'03)*, Asilomar, CA, USA, January 5-8 2003.

[9] M. Egenhofer and D. Mark. Naive geography. In A. Frank and W. Kuhn, editors, *Spatial Information Theory: A Theoretical Basis for GIS*, number 988 in Lecture Notes in Computer Sciences, pages 1–15, Berlin, 1995. Springer-Verlag.

[10] D. P. Fosca Giannotti, Mirco Nanni. Efficient mining of temporally annotated sequences. In *SIAM Data Mining Conf. 2006 (SDM06)*, 2006.

[11] A. Friis-Christensen, C. S. Jensen, J. Nytun, and D. Skogan. A conceptual schema language for the management of multiple representations of geographic entities. In *Transactions in GIS*, volume 9, pages 345–380. 2005.

[12] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns*. Addison-Wesley Professional, January 1995.

[13] R. H. Güting, V. T. Almeida, and Z. Ding. Modeling and querying moving objects in networks. In *VLDB Journal*, volume 15, pages 165–190. 2006.

[14] R. H. Güting, T. Behr, V. Almeida, Z. Ding, F. Hoffmann, and M. Spiekermann. SECONDO: An Extensible DBMS Architecture and Prototype. Informatik Report 313, FernUniversität Hagen, 2004.

[15] R. H. Güting, M. H. Böhlen, M. Erwig, C. S. Jensen, N. A. Lorentzos, M. Schneider, and M. Vazirgiannis. A foundation for representing and quering moving objects. *ACM Transactions on Database System*, 25(1):1–42, 2000.

[16] R. H. Güting and M. Schneider. *Moving Objects Databases*. Diane D. Cerra, 2005.

[17] K. Hornsby and M. J. Egenhofer. Modeling moving objects over multiple granularities. *Ann. Math. Artif. Intell.*, 36(1-2):177–194, 2002.

[18] V. Khatri, S. Ram, and R. Snodgrass. Augmenting a conceptual model with geospatiotemporal annotations,. In *IEEE Transactions on Knowledge and Data Engineering*, volume 16, pages 1324–1338. 2004.

[19] M. Kwan and J. Lee. *Geo-visualization of Human Activity Patterns Using 3-D GIS: A Time-Geographic Approach*, chapter Spatially Integrated Social Science: Examples in Best Practice. Oxford University Press, 2003. Chapter 3.

[20] P. Laube, S. Imfeld, and R. Weibel. Discovering relative motion patterns in groups of moving point objects. In *International Journal of Geographic Information Science*, volume 19, pages 639–668. Taylor & Francis Group, July 2005.

[21] D. Mark, M. Egenhofer, L. Bian, P. Rogerson, and J. Vena. Spatio-temporal GIS Analysis for Environmental Health. In *2nd International Workshop on Geography and Medicine (GEOMED')*, Paris, France, 1999.

[22] V. Noyon, T. Devogele, and C. Claramunt. A relative modelling approach for spatial trajectories. In *Proceedings of the 4th ISPRS Workshop on Dynamic and Multi-dimensional GIS*, University of Glamorgan, Archives of ISPRS, September 5-8 2005.

[23] Oracle. Oracle database 10g. http://www.oracle.com/database/index.html, April 2007.

[24] C. Parent, S. Spaccapietra, and E. Zimanyi. *Conceptual Modeling for Traditional and Spatio-Temporal Applications - The MADS Approach.* Springer, 2006.

[25] C. Parent, S. Spaccapietra, and E. Zimanyi. The MurMur project: Modeling and querying multi-representation spatio-temporal databases. In *Information Systems*, volume 31, pages 733–769. 2006.

[26] M. Pelanis, S. Simonas, and C. S. Jensen. Indexing the past, present and anticipated future positions of moving objects. In *ACM Transactions on Database Systems*, volume 31, pages 255–298. March 2006.

[27] N. Pelekis, Y. Theodoridis, S. Vosinakis, and T. Panayiotopoulos. Hermes - a framework for location-based data management. In *EDBT*, pages 1130–1134, 2006.

[28] N. Pelekis and B. Theodoulidis. TAU TLL Data Cartridge. Technical Report TR-02-3, CRIM, UMIST, Manchester, UK, 2002.

[29] L. Speicys, C. S. Jensen, and A. Kligys. Computational data modeling for network-constrained moving objects. In *ACM International Symposium on Advances in Geographic Information Systems (GIS'03)*, pages 118–15, New Orleans, Lousiana, USA, November 7-8 2003.

[30] M. Theriault, C. Claramunt, A. Seguin, and P. Villeneuve. *Advances in Spatial Data Handling*, chapter Temporal GIS and Statistical Modeling of Personal lifelines, pages 433–449. Springer-Verlag, Heidelberg, New York, 2002.

[31] S. Winter and M. Raubal. Time geography for ad-hoc shared-ride trip planning. In *7th International Conference on Mobile Data Management (MDM'06)*, Nara, Japan, 2006.

[32] O. Wolfson, B. Xu, S. Chamberlain, and L. Jiang. Moving objects databases: Issues and solutions. In *SSDBM '98: Proceedings of the 10th International Conference on Scientific and Statistical Database Management*, pages 111–122, Washington, DC, USA, 1998. IEEE Computer Society.