Types and Operators in M-Atlas system

Roberto Trasarti, Salvo Rinzivillo, Mirco Nanni, Fosca Giannotti
KDD Lab
ISTI-CNR, Pisa

November 7, 2011

Abstract

In this technical report we illustrate the types managed by the M-Atlas system and the operators defined among them. Due to the complexity of spatio-temporal data, models and patterns, the system is be based on a rich formalism, capable to representing the specificity of movement data. We choose the object-relational model, which combines the simplicity of the relational data model and SQL with the basic object oriented capabilities. The main feature of the object- relational database model is that objects and classes are directly supported in database schemas supporting the extension of the original types with custom types representing complex structures. Moreover using a pre-existing GIS technology developed on the database the new types and operators can be integrated easily.

1 Introduction

M-Atlas adopts state-of-the-art moving object database design principles for its trajectory store, extended with mechanisms for managing and querying models and patterns. There are three main object types in M-Atlas: Data, M-model, and M-pattern, depicted in Figure 1. We distinguish between *models* and *patterns*: a pattern is a representation of a local property that holds over a sub-group of mobility data, e.g., a flock of trajectories; on the other hand, a model is a representation of a global property that holds over an entire dataset: accordingly, a model is either a global aggregate (e.g., speed distribution in a trajectory dataset) or a collection of patterns (e.g., the clustering that partitions an entire dataset into separate clusters).

Practically the system adds new object-relational types to the database in order to represent the new types of data, patterns and models. The advantage of having an object-relational representation is threefold: (i) it allows the definition of complex data such as lists and trees, (ii) yields a compact representation of the data, and (iii) makes it possible to use classical indexing techniques already in the database on complex objects. Following we will present the formalism of the data, m-pattern and m-model types describing how they are represented in the ODBMS, the implementation details of the types are reported in Appendix A.

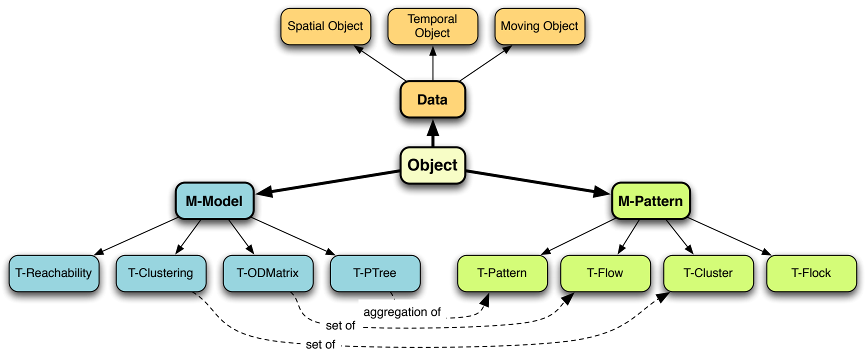


Figure 1: The M-Atlas type hierarchy. M-Model, M-Pattern and Data are the basic types of data. We can notice the relationship between M-Models and M-Patterns. For example, T-Clustering model is represented by a set of T-Cluster patterns, while T-PTree model is an aggregation of T-Patterns

2 Data types

M-Atlas supports three types of data: purely spatial data, purely temporal data, and moving points, or trajectories.
Spatial objects have a geometric shape and a position in space, and are represented as $S = (type, < p_1, \ldots, p_n >)$ where $type \in \{point, line, polygon\}$ defines the meaning of the list of points $< p_1, \ldots, p_n >$: if $type = point$ then the list is composed by only one point with its coordinates; if $type = line$ then the list represents a broken line; if $type = polygon$ then the list represents the contour of the polygon. The representation of this object in the ODBMS coincides with the *geometry* type of the postGIS extension [7] (allowing the compatibility with all the other functionalities given by it).

Temporal objects are represented as $T = (t, d)$ where t is an absolute temporal value (w.r.t. a time reference system) and d is a duration expressed in seconds. When t is equal to the special value *null*, then the temporal object represents a relative time period. An *interval* object is a pair of temporal objects $I = (T_{min}, T_{max})$. This is represented in the ODBMS as a new object containing two attributes: a timestamp and a numeric field.

Moving objects are the spatio-temporal evolution of the position of a spatial object. There are three different types of moving objects: moving point, moving line and moving polygon. In this paper we concentrate on moving points, which represent trajectories. A moving point is defined as $Mo =< p_1, t_1 >, \ldots, < p_n, t_n >$, where p_j is a spatial object representing a point, t_j is a temporal object representing an absolute time point and $t_i < t_j$ for $1 \leq i < j \leq n$. To the purpose of this paper, the terms *trajectory* and *moving point* are synonyms. For representation in the ODBMS the *3D linestring* type is used, where the third dimension of the point is the absolute timepoint expressed in second [1].

Data Constructors can be associated with each data type, allowing, e.g., to construct data objects by acquiring and preprocessing raw data. trajectories need to be created starting from the time-stamped location observations acquired from external sources; such pre-processing requires adequate data acquisition mechanisms tailored to deal with uncertainty, errors and domain-dependent constraints concerning the notions of travel, stop, movement, time resolution, etc. To this purpose, M-Atlas provides a specific constructor operator, that builds a set of trajectories from a table of observations in the form: $< id, x, y, t >$ where *id* is the identifier of the trajectory, x and y are the coordinates of an observation and t the time when the observation is taken. As an example, the following construction query builds a table *Travels* of reconstructed travels from the raw observations contained in the table *RawData*. By setting a maximum space gap (in km) and time gap (in seconds) between any two consecutive observations in a trajectory, we can specify the end of a travel and the beginning of a new one.

```
CREATE DATA Travels BUILDING MOVING_POINTS
 FROM (SELECT userid,lon,lat,datetime FROM RawData
       ORDER BY userid,datetime)
 SET MOVING_POINT.MAX_SPACE_GAP = 0.2 AND
     MOVING_POINT.MAX_TIME_GAP = 1800
```

Other data constructors are provided to build spatial objects, such as rectangles and rectangular grids, and temporal objects, such as intervals and series of consecutive intervals. Although simple data objects, spatio-temporal grids are important ingredients in the analysis of movements.

3 M-Pattern Types

A mobility pattern, M-Pattern in short, represents the common behavior of a (sub-)group of trajectories, obtained as a result of a data mining algorithm. The types of M-Patterns currently supported by M-Atlas are shown in Figure 2.

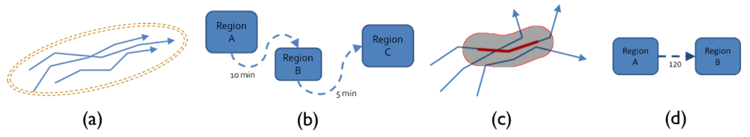


Figure 2: M-Pattern types: (a) T-Cluster, (b) T-Pattern, (c) T-Flock, (d) T-Flow

T-Cluster. A T-Cluster (Figure 2(a)) is defined as a set $S = \{(\tau_1, l), (\tau_2, l), \ldots\}$ of labelled trajectories, which share the same membership tag l. The trajectories of a T-Cluster are grouped on the basis of their similarity according to a specified similarity function, chosen from a repertoire of possible choices. In the ODBMS we don't replicate the data to represent the list of moving points, hence the object is composed by a

[1]using the standard transformation which consider as base date the 01/01/1970

list of ids of moving points and the a text field which is the name of the function (in the M-Atlas system library) which is used to compute it.

T-Pattern: it is represented as $tp = (R, T, s)$ where $R =< r_0, \ldots, r_k >$ is a sequence of regions, $T =< t_1, \ldots, t_k >$ is a sequence of relative time intervals $t_j = [t_j^s, t_j^e]$ associated to each region and s is the support of tp, i.e., the number of trajectories that are compatible with tp in space and time. Informally, a T-Pattern can be represented as $r_0 \xrightarrow{t_1} r_1 \cdots \xrightarrow{t_k} r_k$. Originally introduced in [4], a T-Pattern (Figure 2(b)) is a concise description of frequent behaviors, in terms of both space (i.e., the regions of space visited during movements) and time (i.e., the duration of movements). In the ODBMS we represent the time intervals as a new object which contains two temporal objects and we use a spatial objects for the regions, therefore the representation of the whole T-Pattern is an object by a list of regions, a list of intervals and a numeric value for the support.

T-Flock. A T-Flock $f = (I, r, b)$ represents a spatio-temporal coincidence of a group of moving points, where $I = [t_{min}, t_{max}]$ is the time interval of the coincidence, b is the base moving point and r is the spatial buffer around b which is used to determine the coincidence. This spatio temporal coincidence defines a common behavior of the people which move together for a certain time interval (Figure 2(c)). In the database we represent it as an object composed by an interval of time (using the same representation of it introduced for the T-Pattern), a numeric value for the radius and a moving point for the base.

T-Flow. The T-Flow $tf =< R_1, R_2, w >$ represents a flow of $w \geq 0$ trajectories which move from region R_1 to region R_2 (Figure 2(d)). In the ODBMS we represent a single flow as a row in a table which containing the two spatial objects and a numeric attribute for the size of the flow.

4 M-Model Types

Mobility models, M-Models in short, are the global models extracted by a data mining algorithm, where the adjective *global* indicates the fact that each such model describes the entire input dataset. Figure 3 illustrates some of the available M-models in M-Atlas; other M-Models are simply the entire collection of T-Patterns, T-Clusters and T-Flocks mined over a trajectory dataset.

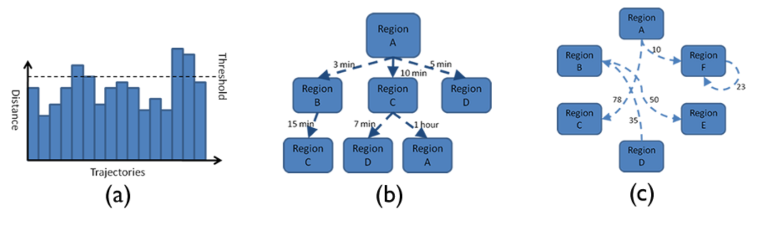


Figure 3: M-Models types: (a) Reachability plot, (b) T-PTree and (c) T-ODMatrix.

Reachability plot: is a histogram of distances between trajectories, obtained considering a specific distance function (Figure 3(a)). More precisely, it is a sequence of pairs $Rp =< (t_1, d_1) \ldots (t_n, d_n)) >$ where t_j is a trajectory and d_j is the distance between t_j and t_{j+1}, where t_{j+1} is the nearest neighbor of t_j which does not occur in $\{t_1, \ldots, t_j\}$. Using a threshold ϵ for distance, the reachability plot identifies a set of T-Clusters representing the partition of the whole dataset into labelled groups of similar trajectories. This model is represented in the database as an object containing a list of moving object ids (to avoid data replication) with associated to their distances.

T-PTree. A T-Pattern Tree, T-PTree in short, is a compact representation of a set of T-Patterns (Figure 3(b)). It is a prefix tree $PT = \{root, N, E\}$, where N is the set of nodes of the tree, E is the set of edges and $root$ is the root of the tree. Each node $n_i = \{r, supp\}$ contains a spatial region r and a support value $supp$; each edge $e_{i,j} = \{t_{min}, t_{max}\}$ connects the nodes i and j specifying a relative time interval. The support label on the nodes represent the maximum support value of the T-Patterns which have the path $root, \ldots, n_i$ as prefix. The formal definition of prefix of a T-Pattern is in [3]; intuitively a T-pattern tp_1 is prefix of another T-Pattern tp_2 if every region and interval of the first pattern are included in the region and interval of the second, in the specified order. In the ODBMS we represent the pattern tree in a recursive way: a node is described by a spatial region, a numeric value for the support, a list of its children nodes and a list of time intervals describing the edges labels. With this representation the root of the tree is a simple node containing the whole tree.

T-O/DMatrix. A T-O/DMatrix (Figure 3(c)) is defined as a labeled graph $odm = \{O, D, E\}$ where $O = \{o_1 \ldots o_n\}$ are the nodes which identify the origins, $D = \{d_1 \ldots d_k\}$ are the nodes which identify the destinations and E are the edges which connect an origin node with a destination node. Each node (both origins and destinations) contains a spatial region and the label on the edges represent the number of movements which start in the origin region and end in destination node. This model results from the composition of a set of T-Flows, each representing the trajectories from the origin to the destination region. In the ODBMS we represent the T-ODMatrix as a table of flows, hence this model id represented as a table of rows containing the two spatial objects and a numeric value as the number of trajectories which move between the two.

Model and Pattern constructors: a generic constructor for M-Models (and M-Patterns) is defined as a function $T_d \rightarrow (T_m, T_p)$ where T_d is a data table, T_m is a model table (containing a single M-Model object) and T_p is a table containing a set of M-Patterns objects. This operator realizes the construction of M-Models and M-Patterns through the execution of a data mining method with a specified parameter setting. M-Atlas provides a mining constructor for each method in its *data mining library*, more details are presented in [5] and [6]. An example of mining constructor query is the following, which generates a step of density-based trajectory clusters under specific parameters:

```
CREATE MODEL ClusteringTable MINE AS T-CLUSTERING
FROM (Select t.id, t.trajobj from TrajectoryTable t)
SET T-CLUSTERING.FUNCTION = ROUTE_SIMILARITY AND
    T-CLUSTERING.EPS = 100 AND
```

```
T-CLUSTERING.MIN_PTS = 20
```

5 Spatio-temporal query primitives

The querying primitives over data, models and patterns are summarized in Figure 4; the upper left square contains the *data* $\times$ *data* primitives, corresponding to the classical spatio-temporal primitives defined in [1]. All the other primitives have been specifically designed for M-Atlas, in that they involve models and patterns (*data* $\times$ *model/pattern*, *model/pattern* $\times$ *data* or *model/pattern* $\times$ *model/pattern*).

Each primitive is defined as a function $r(T_1, T_2) \rightarrow (T_{rel})$, where T_1 and T_2 are two sets of objects and $T_{rel} = \{\langle o_1, o_2 \rangle | o_1 \in T_1 \wedge o_2 \in T_2 \wedge rel(o_1, o_2)\}$. Here, rel is a predicate defined between the types of objects in T_1 and T_2, which specifies the relation that should hold over the pairs of objects that are kept in the resulting table T_{rel}.

Albeit there are apparently only a few kinds of spatio-temporal primitives (*contains*, *intersects*, *equals*), a large variety comes from the different combinations of types of objects to which such primitives are applied, as illustrated in Figure 4. Each combination depends on the semantics of movement represented by the types of the involved objects; for instance, the definition of *intersects* between a T-pattern and a Moving Point is completely different from that between a T-Flock and a Moving point. The expressive power of M-Atlas derives exactly from the comprehensive repertoire of spatio-temporal primitives over all combinations of data, patterns and models; the entire repertoire is reported in [6].

A *pattern* $\times$ *pattern* primitive is the *contains* relation between two T-Patterns $tp^1 = (R^1, T^1, s^1)$ and $tp^2 = (R^2, T^2, s^2)$, defined as follows:

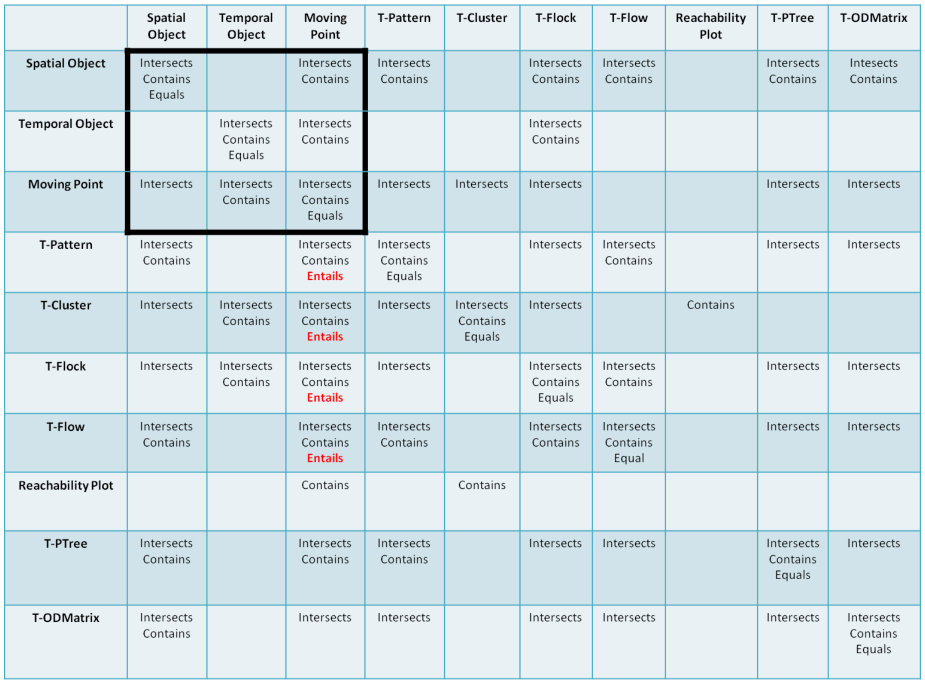
$$contains(tp^1, tp^2) \equiv \exists k > 0 \mid contains(R^1_k, R^2_k) \wedge \ldots \wedge contains(R^1_{k+n}, R^2_{k+n}) \wedge contains(T^1_k, T^2_k) \wedge \ldots \wedge contains(T^1_{k+n}, T^2_{k+n}), n = |R^2|$$

where the *contains* operator between regions and temporal intervals (*data* $\times$ *data*) is defined as in [1]. To construct the table of pairs of objects that satisfy a generic relation we use the query syntax `CREATE RELATION`, as in the following example, where a table of pairs of T-patterns (tp_1, tp_2) is created, such that tp_1 contains tp_2:

```
CREATE RELATION TPatternContains USING CONTAINS
 FROM (SELECT t1.id, t1.tpattern, t2.id, t2.tpattern
       FROM TPatternTable t1, TPatternTable t2
       WHERE t1.id <> t2.id)
```

A distinctive *pattern* $\times$ *data* primitive is the *entails* relation. $entails(p, o)$ holds if the data object o is an instance of pattern p. The definition of *entails* is specific for each M-Pattern, and details are given in Sec. **??**. An example of query is the following, which creates a table containing the trajectories belonging to a specific T-Cluster:

```
CREATE RELATION TrajectoriesInCluster USING ENTAILS
FROM (SELECT t.id, t.traj, c.id, c.cluster
      FROM TrajectoryTable t, ClustersTable c)
```



	Spatial Object	Temporal Object	Moving Point	T-Pattern	T-Cluster	T-Flock	T-Flow	Reachability Plot	T-PTree	T-ODMatrix
Spatial Object	Intersects Contains Equals		Intersects Contains	Intersects Contains		Intersects Contains	Intersects Contains		Intersects Contains	Intesects Contains
Temporal Object		Intersects Contains Equals	Intersects Contains			Intersects Contains				
Moving Point	Intersects	Intersects Contains	Intersects Contains Equals	Intersects	Intersects	Intersects			Intersects	Intersects
T-Pattern	Intersects Contains		Intersects Contains Entails	Intersects Contains Equals		Intersects	Intersects Contains		Intersects	Intersects
T-Cluster	Intersects	Intersects Contains	Intersects Contains Entails	Intersects	Intersects Contains Equals	Intersects		Contains		
T-Flock	Intersects	Intersects Contains	Intersects Contains Entails	Intersects		Intersects Contains Equals	Intersects Contains		Intersects	Intersects
T-Flow	Intersects Contains		Intersects Contains Entails	Intersects Contains		Intersects Contains	Intersects Contains Equal		Intersects	Intersects
Reachability Plot			Contains		Contains					
T-PTree	Intersects Contains		Intersects Contains	Intersects Contains		Intersects	Intersects		Intersects Contains Equals	Intersects
T-ODMatrix	Intersects Contains		Intersects	Intersects		Intersects	Intersects		Intersects	Intersects Contains Equals

Figure 4: M-Atlas spatio-temporal primitives

Transformation primitives. Transformations are a class of primitives which uses external methods to perform complex data pre-processing and model/pattern post-processing operations in the knowledge discovery process. Among the many transformations available in M-Atlas, *T-Anonymization*, transforms a set of trajectories into a new set satisfying K-anonymity, i.e., one where each trajectory is indistinguishable from at least other K similar trajectories [2]. An example of transformation query that pre-processes data for anonymity is the following, where the parameter *K* sets the minimum anonymity threshold to be guaranteed.

```
CREATE TRANSFORMATION AnonymizedData USING ANONYMIZATION
 FROM (SELECT t.id, t.trajobj FROM TrajectoryTable t)
 SET ANONYMIZATION.K = 10
```

The complete grammar of the DMQL is reported in appendix B.

6 M-Atlas architecture

The architecture of M-Atlas is composed of two main components: the *Graphical User Interface*, supporting the visual analytic process, and the *M-Atlas Engine*, providing the full power of the data mining query language.

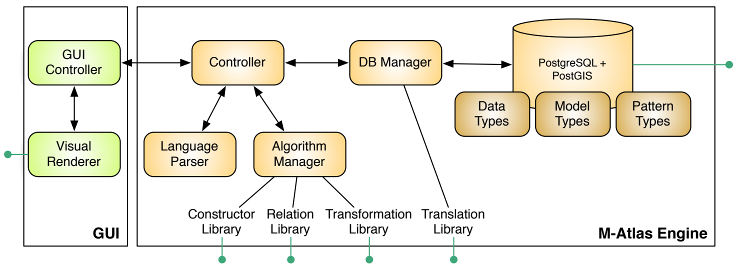


Figure 5: M-Atlas system architecture.

In see Figure 5 we provide a schema of the system architecture: a query is submitted through the graphical interface to the *Controller* module, which coordinates the tasks performed by all other modules. The *Language Parser* analyzes the input query. Standard SQL queries are directly sent to the *Database Manager* and executed by the *Object Relational DBMS*. All other M-Atlas queries are translated by the *Language Parser* into an execution plan, which combines both DB queries and calls to the methods provided by the *Algorithm Manager*. The results of a query is stored into the *ORDBMS* and possibly displayed, through the *Controller*, by the *Graphical User Interface*. The pins represent the modules which can be extended by the plug-in system.

The architecture has been designed as a plug-in environment, where new models and patterns can be easily added, together with their mining algorithms. Extending the system requires four steps: *(i)* the new model/pattern type is introduced in the DB; *(ii)* the Translation Library of the DB Manager is extended with the access methods for the new type; *(iii)* the mining method associated with the new type is added to the Constructor Library; *(iv)* the spatio-temporal primitives associated with the new type are added to the Relation Library. M-Atlas is being continuously extended with new functionalities. A basic requirement for the architecture is minimizing memory usage during query execution. To this purpose, query results are managed, as far as possible, by reference in streaming fashion, i.e., by processing iteratively one set of rows of fixed size at a time, both during loading and storing. However, the system adapts to the memory policy of the various mining algorithms. Therefore, the memory consumption of most M-Atlas queries is constant, with the remarkable exception of the mining algorithms, which require multiple passes over data.

7 Conclusions

We can summarize the advantages of having the presented formalism for representing data, models and pattern and a data mining query language with the following features:

1. *the compositionality of the operators* allows the user to create their own knowledge discovery process combining the different operators;

2. *the iterative querying* capability allows the user to apply the data mining algo-

rithms on the data to discover patterns and models, but also apply such patterns and models to the data for a deeper analysis. This is an iterative process which allows the user to use the models not only as static knowledge to be presented as result, but as an active element of the process used to go deeper in the data understanding.

3. *the repeatability of the process*: having a language that supports the steps of the discovery process allows to materialize the executed process *as a language script*. Thus, the output is not only the set of mined patterns, but also the script storing the process thus making the process repeatable on different datasets.

A full set of applications showing the capability of the language and the system is presented in [5].

References

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

[2] Abul, O., Bonchi, F. and Nanni, M. $\mathcal{N}$ever $\mathcal{W}$alk $\mathcal{A}$lone: Uncertainty for anonymity in moving objects databases. In *Proc. of the 24nd IEEE Int. Conf. on Data Engineering (ICDE'08)*, 2008.

[3] Monreale, A. Pinelli, F. Trasarti, R, and Giannotti F. Wherenext: a location predictor on trajectory pattern mining. In *15th ACM SIGKDD Conference on Knoledge Discovery and Data Mining (KDD'09)*, 2009.

[4] Giannotti F., Nanni M. Pinelli F., and Pedreschi D. Trajectory pattern mining. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining* , pages 330–339, 2007.

[5] Giannotti F, Nanni M, Pedreschi D., Pinelli F., Renso C., Rinzivillo S. and Trasarti R. Unveiling the complexity of human mobility by querying and mining massive trajectory data. In VLDB Journal Special issue on Data Management for Mobile Services (2011).

[6] Trasarti, R. *Mastering the Spatio-Temporal Knowledge Discovery Process*. PhD in Computer science, University of Pisa, 2010.

[7] http://www.postgis.org/docs/reference.html

A Appendix: Types signatures

Here the signatures of the data, m-pattern and m-model data types are presented.

A.1 Data type signatures

- The spatial object data type is implemented using the GEOMETRY type of PostGis Extension [7].

- The temporal object type is defined as:

  ```
  CREATE TYPE period AS (
      timepoint timestamp,
      duration double precision
  );
  ```

- The moving object data type is implemented using the LINESTRING type of PostGis Extension [7].

A.2 M-Pattern type signatures

- the T-Cluster object type is defined as:

  ```
  CREATE TYPE cluster AS (
      elements text[],
  );
  ```

- the T-Pattern object type is defined as:

  ```
  CREATE TYPE tpattern AS (
  regions text[],
  min_interval double precision[],
  max_interval double precision[],
  support numeric
  );
  ```

- the T-Flock object type is defined as:

  ```
  CREATE TYPE flock AS (
      elements text[],
      base text
  );
  ```

- The T-Flow is represented by a row in a relational table with three attributes:

  ```
  from_region geometry,
  to_region geometry,
  num numeric
  ```

A.3 M-Model type signatures

- The reachability plot is represented as list of reachability plot entries defined as:

  ```
  CREATE TYPE reach_plot_entry AS (
      obj_id text,
      rd numeric,
      cd numeric,
  );

  create TYPE reach_plot as(
     entries reach_plot_entry[]
  );
  ```

- The TP-Tree object type is defined as:

  ```
  CREATE TYPE TP-Tree AS (
     obj_id text,
     parent_id text,
     children_id text[],
     edges_label period[],
     region geometry,
     support numeric,
   );
  ```

- The T-O/D Matrix is represented by a table of T-Flow, hence the definition of the table is the following:

  ```
  CREATE TABLE od_matrix AS (
     from_region geometry,
     to_region geometry,
     num numeric
  );
  ```

B Appendix: DMQL grammar

The data mining query language provided by the M-Atlas system is able to define a complete knowledge discovery process trough a script. following we present the formal grammar of the DMQL:

```
DMQLCall:=
   BuildCall |
   RelationCall |
   TransfCall |
   MiningCall |
   SqlCall
```

```
BuildCall:= CREATE  OBJECT TableName AS BUILD DataType
   FROM ( SqlCall ) [WHERE [ParamBuilder [AND] ]*]
RelationCall:= CREATE RELATION TableName USING RelationType
   FROM ( SqlCall )
TransfCall:= CREATE TRANSFORMATION TableName USING TransfType
   FROM ( SqlCall ) [WHERE [ParamTransf [AND] ]*]
MiningCall:= CREATE MODEL TableName AS MINE MiningAlgorith
   FROM ( SqlCall ) [WHERE  [ParamAlgorithm [AND] ]*]
SqlCall:= [SQL standard]

TableName:= [A-Z, 0-9]*
DataType:= MOVING_POINT |
   GEOMETRY |
   PERIOD
RelationType:= CONTAINS |
   INTERSECTS |
   EQUALS |
   ENTAILS
TransfType:= INTERSECTION |
   ANONYMIZATION |
   RESAMPLING |
   [...]
MiningAlgorithm:= T-PATTERN |
   OPTICS  |
   FLOCK |
   [...]
ParamTransf:= TransformationType . Parameter = Value
ParamBuilder:= DataType . Parameter = Value
ParamAlgorithm:= MiningAlgorithm . Parameter = Value
Parameter := [A-Z, 0-9]*
Value := [A-Z, 0-9]*
```

Where [SQL standard] represent the part of grammar where the standard SQL is recognized and [...] represent the fact that some algorithm and transformation are omitted.