Deductive and inductive reasoning on spatio-temporal data

Mirco Nanni[1], Alessandra Raffaetà[2], Chiara Renso[1], Franco Turini[3]

[1] ISTI CNR - Pisa
{nanni, renso}@isti.cnr.it
[2] Dipartimento di Informatica - Università Ca' Foscari Venezia
raffaeta@dsi.unive.it
[3] Dipartimento di Informatica - Università di Pisa
turini@di.unipi.it

Abstract. We present a framework for a declarative approach to spatio-temporal reasoning on geographical data, based on the constraint logical language STACLP, which offers deductive and inductive capabilities. It can be exploited for a deductive rule-based approach to represent domain knowledge on data. Furthermore it is well suited to model trajectories of moving objects, which can be analysed by using inductive techniques, like clustering, in order to find common movement patterns. A sketch of a case study on behavioral ecology is presented.

1 Introduction

New technologies in the field of mobile computing and communication can provide a wealth of spatio-temporal information. Collected data are useful as far as they can be used to analyse phenomena and to take informed decisions. The first step for allowing one to make a profitable use of data is to provide a query language for them, and we believe that the query language, even more in the case of spatio-temporal data, must be able to handle not only data but also rules, and exhibit both deductive and inductive capabilities. Rules can be used to represent general knowledge about the collected data, and deductive capabilities can provide answers to queries that require some inference besides the crude manipulation of the data. Induction can help extracting implicit knowledge from data and, according to the impressive success in the knowledge discovery in database field, is a powerful support to decision making.

In order to support both deductive and inductive reasoning on spatio-temporal data we propose the formalism STACLP (Spatio-Temporal Annotated Constraint Logic Programming) [11] based on constraint logic programming, extended with annotations. Constraint logic programming provides the deductive capabilities, and annotations allow a neat representation of temporal, spatial and spatio-temporal knowledge. On this ground we can construct the inductive capabilities of the language by implementing knowledge extraction methods.

Our proposal fits in a new and promising research field, that is the integration of declarative paradigms and systems for dealing with spatial and/or temporal information, such as spatial databases and Geographical Information Systems (GISs). In the literature we can find other attempts to exploit the deductive capabilities of logics to *reason* on geographic data [14, 13, 1, 8].

Information induction over spatio-temporal data is mainly an unexplored field, yet. In this context, [2] suggests two main kinds of information to induce: *meta-rules*, i.e., regularities shown along time by the rules obtained in each snapshot, and *evolution rules*, i.e., rules computed over pre-computed spatio-temporal features of entities. So far, very little attention has been paid to the mining of spatio-temporal classifiers, association rules and time sequences, while a few proposals exist for spatio-temporal clustering [9, 7, 5], which go beyond the trivial extensions of spatial clustering methods.

Section 2 introduces the language STACLP and Section 3 focuses on the representation of one of the most fundamental spatio-temporal objects, i.e. the trajectory. In Section 4 we sketch how STACLP allows us to solve a case study concerning behavioural ecology by using the deductive capabilities of the language, and the inductive capabilities implementing a clustering method. Finally, Section 5 draws some conclusions and outlines future research directions.

2 STACLP: a Spatio-Temporal Language

In this section we present the language STACLP [11], which extends Temporal Annotated Constraint Logic Programming [6], a constraint logic programming language with temporal annotations, by adding spatial annotations. The pieces of spatio-temporal information are given by pairs of annotations which specify the spatial extent of an object at a certain time period. The use of annotations makes time and space explicit but avoids the proliferation of spatial and temporal variables and quantifiers. Moreover, it supports both definite and indefinite spatial and temporal information, and it allows one to establish a dependency between space and time, thus permitting to model continuously moving points and regions.

For space limitation we can only give an informal description of the language and we refer the reader to [11] for more details and for a formal definition of the semantics of STACLP programs. Let us start by describing the temporal and spatial domain underlying STACLP. Time can be discrete or dense. Time points are totally ordered by the relation $\leq$. We denote by $\mathcal{T}$ the set of time points and we suppose to have a set of operations (e.g., the binary operations $+$, $-$) to manage such points. The time-line is left-bounded by 0 and open to the future, with the symbol ∞ used to denote a time point that is later than any other. A *time period* is an interval $[r, s]$ with $r, s \in \mathcal{T}$ and $0 \leq r \leq s \leq \infty$, which represents the convex, non-empty set of time points $\{t \mid r \leq t \leq s\}$. Thus the interval $[0, \infty]$ denotes the whole time line.

Analogously space can be discrete or dense and we consider as spatial regions *rectangles* represented as $[(x_1, x_2), (y_1, y_2)]$, where (x_1, y_1) and (x_2, y_2) denote the lower-left and upper-right vertex of the rectangle. Precisely, $[(x_1, x_2), (y_1, y_2)]$

models the region $\{(x,y) \mid x_1 \leq x \leq x_2, y_1 \leq y \leq y_2\}$. Rectangles are the two-dimensional counterpart of convex sets of time points.

Annotated formulae are of the form $A\,\alpha$ where A is an atomic formula and α an annotation. We define three kinds of temporal and spatial annotations inspired by similar principles:

- $\mathtt{at}\,T$ and $\mathtt{atp}\,(X,Y)$ are used to express that a formula holds in a time or spatial point, respectively.
- $\mathtt{th}\,I$, $\mathtt{thr}\,R$ are used to express that a formula holds *throughout*, i.e., at *every* point, in the temporal interval or the spatial region, respectively.
- $\mathtt{in}\,I$, $\mathtt{inr}\,R$ are used to express that a formula holds at *some* point(s) - but we may not know exactly which - in the interval or the region, respectively. They account for indefinite information.

The set of annotations is endowed with a partial order relation $\sqsubseteq$ on the information content: given two annotations α and β, the intuition is that $\alpha \sqsubseteq \beta$ if α is "less informative" than β in the sense that for all formulae A, $A\,\beta \Rightarrow A\,\alpha$. For instance, for temporal annotations, $\mathtt{th}\,[s_1,s_2] \sqsubseteq \mathtt{th}\,[r_1,r_2]$ iff $[s_1,s_2] \subseteq [r_1,r_2]$, whereas $\mathtt{in}\,[s_1,s_2] \sqsubseteq \mathtt{in}\,[r_1,r_2]$ iff $[s_1,s_2] \supseteq [r_1,r_2]$. Spatial annotations are treated analogously. Spatio-temporal annotations are pairs of spatial and temporal annotations, with the componentwise order.

The partial order on annotations is used in the definition of two new inference rules in addition to Modus Ponens that increase the expressive power of the language. The first one states that if a formula holds with some annotation, then it also holds with "less informative" (smaller w.r.t. $\sqsubseteq$) annotations. The other rule states that if a formula holds with two annotations then it holds also with their least upper bound.

Now we can introduce the clausal fragment of STACLP, which can be used as an efficient spatio-temporal programming language. It consists of clauses of the following form:

$$A\,\alpha\beta \leftarrow C_1,\ldots,C_n, B_1\,\alpha_1\beta_1,\ldots,B_m\,\alpha_m\beta_m \quad (n,m \geq 0)$$

where A is an atom, α, α_i, β, β_i are (optional) temporal and spatial annotations, the C_j's are constraints and the B_i's are atomic formulae. Constraints C_j cannot be annotated. A *STACLP program* is a finite set of STACLP clauses.

Example 1. Assume that a person is described by his/her name, the activity and the spatial position(s) in a *certain time interval*. For instance, from 1am to 10am John sleeps, then he goes skiing up to 4pm, while Monica skies from noon to 4pm. This can be expressed by means of the following clauses.

```
does(john,sleep) atp (200,700) th [1am,10am].
does(john,ski) inr [(500,2000),(1000,2000)] th [10am,4pm].
does(monica,ski) inr [(500,800),(800,2000)] th [12am,4pm].
place(ski_shop) thr [(700,900),(1200,1400)].
```

Notice that the spatial location is expressed by using an `atp` annotation when the exact position is known, and by an `inr` annotation if we can only delimit the area where the person can be found. Furthermore, a place can be described by its name and its area represented by a `thr` annotation.

An example of query is *Where is John while Monica is at the ski shop?*, encoded in the following way: `does(john,_) inr R th I, does(monica,_) inr R1 th I, place(ski_shop) thr R1`. This query is a composition of a spatial join and a temporal join.

3 Representing Trajectories

One of the basic forms of spatio-temporal information is given by spatio-temporal objects, namely objects which move along time within a spatial environment.

From an abstract point of view, the movement of a spatio-temporal object o – i.e., its *trajectory* – can be represented by a continuous function of time which, given a time instant t, returns the position at time t of the object in a d-dimensional space (typically $d \in \{2, 3\}$). Formally $o : \mathbb{R}^+ \rightarrow \mathbb{R}^d$.

In a real-world application, however, object movements are given by means of a finite set of *observations* – or *control points* –, i.e. a finite subset of points taken from the actual continuous trajectory. Moreover, it is reasonable to expect that observations are taken at irregular rates within each object, and that there is not any temporal alignment between the observations of different objects. As a result, it is possible to have couples of objects for which at all time points the observation of at least one of the objects is missing. A very basic operation such as the comparison between objects, then, cannot be performed by simply comparing their raw observations. To allow the comparison between objects, an (approximate) reconstruction of their full trajectory is needed. Among the several possible solutions, we will focus on *Local interpolation*.

According to the local interpolation method, although there is not a global function describing the whole trajectory, objects are assumed to move between the observed points following some rule. For instance, a linear interpolation function models a straight movement with constant speed, while other polynomial interpolations can represent smooth changes of direction. The above mentioned linear (local) interpolation, in particular, seems to be a quite standard approach to the problem (see, e.g., Chomicki and Revesz's parametric 2-spaghetti model [4]).

Now, let us show how trajectories can be modelled in STACLP. Given an object o, the *observations* which describe its trajectory is a finite set of triplets (x, y, t), where x and y are the coordinates of the object tracked at time t. The trajectory reconstruction through linear interpolation can be easily represented by equations of the following form, which define the (x, y) coordinates of an object at time $t \in [t_1, t_2]$: $(x - x_1)(t_2 - t_1) - (x_2 - x_1)(t - t_1) = 0$, and analogously for the y coordinate, (x_1, y_1, t_1) and (x_2, y_2, t_2) being two consecutive localisation points (i.e., there are not other localisations in the time interval $[t_1, t_2]$).

Specifically, the N localisations of each object o, (x_i, y_i, t_i) for $i = 1, \ldots, N$, can be represented by the following N STACLP facts:

```
fix(o) atp (x1, y1) at t1.     ...     fix(o) atp (xN, yN) at tN.
```

Such localisations will define the *core* of the trajectory of object o, which is then *completed* by defining all the intermediate points through linear interpolation using the following STACLP rules:

```
traj(O) atp (X, X) at T :- fix(O) atp (X, Y) at T.
traj(O) atp (X, Y) at T :- fix(O) atp (X1, Y1) at T1,
                           fix(O) atp (X2, Y2) at T2,
                           succ(T1,T2), T1 < T < T2,
                           X=(X1(T2-T)+X2(T-T1))/(T2-T1),
                           Y=(Y1(T2-T)+Y2(T-T1))/(T2-T1).
```

In the body of the second rule, approximated points (x, y) are computed by using the equation for the line passing through two given points, shown above. The presence of the (standard) *successor* predicate `succ`, defined as true for all and only the couples of (strictly) consecutive localisation points, ensures that no other observation exists between times t_1 and t_2, i.e., the interpolation is performed only between consecutive localisation points.

4 Spatio-Temporal Analysis in STACLP

In this section we sketch a case study about behavioural ecology, the science which studies animal behaviour with special interest in the relation to the environment where animal lives. This is definitely an interesting application domain for our framework since the problems coped with require the analysis of large spatio-temporal datasets.

Typically, biologists collect information tracking the movement of animals by means of special radio-collars thus building large datasets containing spatio-temporal localisations, called *fixes*. Each fix includes the identifier of the animal, the position expressed by the spatial coordinates X, Y and the time T of the localisation. The set of fixes allows us to view animals as spatio-temporal objects.

To experiment the usefulness of our framework for spatio-temporal analysis, we coped with some relevant problems in the behavioural ecology of the crested porcupine [3], such as determining the estimated position of the den whenever its real location is unknown, detecting how the life area of the animal, called *home range*, changes along the time or discovering the relationships existing among individuals. Here we will describe only the last problem. It includes a wide range of cases such as finding pairs of individuals of different sex that move together and possibly share the same den (couples), or groups of individuals that move together (herds) or couple/groups of individuals that avoid each other (territoriality).

In facing such issues, we can highlight the benefits of STACLP. First of all, the language allows for a high level representation and manipulation of time as well as space thus providing primitive support for reasoning on spatio-temporal data. Secondly, it allows us to mix inductive and deductive steps to perform complex kinds of analysis on the behaviour of crested porcupines.

Section 4.1 below focuses on deduction showing how the deductive capability of our language allows us to discover the animals which are likely to be a couple. Section 4.2, instead, is devoted to introduce mechanisms to support inductive analysis, describing how these tools can be successfully used in our case study.

4.1 Deductive Analysis in STACLP

To model the spatio-temporal localisations of each crested porcupine we define a collection of facts of the kind:

```
fix(id) atp (x,y) at t.
```

specifying the position `x,y`, and the time `t` (expressed in seconds) of a localisation for the animal `id`.

Below we will show the STACLP code that implements the expert criteria by which we successfully solve the question of interest. The rules are slightly simplified by removing some implementation details to focus on the knowledge representation ability of the language. Furthermore the code can be made executable by a simple precompilation step. The rules extensively use the Prolog meta-predicate `findall(X,G,L)` which computes the list `L` of elements `X` that satisfy the goal `G`.

We focus here on the specific problem of finding possible couples, that is animals of different sex that move together. In order to find out the probable couples we exploit the notion of *contemporary* fixes: two fixes are contemporary if they refer to localisations of animals in the same place and at the same time, i.e., we consider a kind of spatio-temporal closeness among individuals. Since the tracking technique usually presents several sources of error, in the analysis two fixes are assumed to be *contemporary* if they fall within a given time interval and the corresponding positions are within a certain distance. The effective values for the temporal and spatial thresholds are established by the domain experts.

Analysing the inter-individual distance between animals obtained by computing the number of contemporary fixes for a pair of animals, the expert can decide whether the pair is a couple or not.

```
couple_in_day(Id1,Id2,R,S,N) at T :-
    findall(c(Id1,Id2), (fix(Id1) atp(X1,Y1) at T1,
                         fix(Id2) atp(X2,Y2) at T2,
                         sex(Id1, S1), sex(Id2, S2), S1 != S2,
                         contem(X1,Y1,X2,Y2,R,S,T1,T2) at T),
           L),
    length(L,N).

contem(X1,Y1,X2,Y2,Rad,Sec,T1,T2) at T:- in_day(T1,T2) at T,
               dist(X1,Y1,X2,Y2,D), D < Rad, abs(T2-T1) < Sec.

couple(Id1,Id2,R,S,Ratio) in [T1,T2] :-
                   couple_in_day(Id1,Id2,R,S,N) in [T1,T2],
                   couple_in_day(Id1,Id2,1000000,S,M) in [T1,T2],
                   Ratio is (N/M).
```

The predicate `couple_in_day` returns the number `N` of contemporary fixes in a day for the pair of crested porcupines `Id1,Id2`. Two fixes are considered contemporary if their spatial and temporal distance is bounded by `R` and `S` respectively, as encoded by the predicate `contem`. The atom `in_day(T1,T2) at T` checks whether either `T1` or `T2` is within the day `T`. Finally, the predicate `couple` returns the ratio between the number of contemporary fixes of the crested porcupines `Id1,Id2` and the number of observations of `Id1,Id2` within `S` seconds at arbitrary distance (concretely this is obtained by setting a very large bound for the distance parameter) in a certain time period.

4.2 Inductive Analysis in STACLP

As shown in the previous subsection, deductive reasoning can be useful to solve analysis problems which essentially require to find entities and values having some, possibly complex, properties.

However, dealing with sophisticated analysis tasks, it is quite common to meet concepts and abstract entities whose definition through deductive rules can be extremely difficult. In many cases, a suitable solution to the problem at hand requires the *extrapolation* of new pieces of information from those already available. In other words, knowledge induction capabilities can be needed to properly tackle some difficult problems.

For this reason, the STACLP language can be fruitfully extended with induction capabilities, such as data mining algorithms. In this section we show (i) how a basic data mining tool, the k-means clustering algorithm, specifically tailored around trajectories, can be defined as STACLP rules, and (ii) how it can be used to provide an alternative solution the problem of discovering couples and herds.

Clustering The clustering task is aimed at identifying clusters embedded in the data, i.e. to partition (although not necessarily in a crisp way) the dataset into collections of data objects, such that within each partition the objects are "similar" to one another, while they are "different" from the objects contained in other partitions.

Among the classical clustering algorithms, K-means is one of the best known and widely used, for its simplicity and its low computational complexity. It is a centre-based algorithm, meaning that clusters are represented by means of artificial objects (the *centres* or *representatives*) which summarise the properties of all the objects in their cluster. The k-means algorithm is essentially an iterative convergence process which tries to find "stable" centres: it starts with k random centres, and then, at each iteration (i) each object is associated with the closest centre, and (ii) new centres are computed. The algorithm ends when the centres are stable, i.e. they do not change any more from an iteration to the next one.

The general k-means clustering schema can be instantiated to a specific k-means algorithm by specifying the two key operations used in the schema: (i) computing the distance between two objects, and (ii) computing the representative of a set of objects (i.e., the centre of a cluster). Different definitions for these two steps can yield completely different notions of clustering.

The STACLP language allows to implement a k-means algorithm in a very compact, well structured and readable form. In what follows, we show the most high level rules of such implementation. For ease of presentation, we assume that `k` is the (fixed) number of clusters to find, and all objects to be clustered have an Id of the form "`objs(name_of_object)`".

```
assignments(A) :- objs_to_cluster(O_list), assign(max_n_iters, A, O_list).
objs_to_cluster(O_list) :-
              findall(X, (fix(X) atp (_,_) at _, X=objs(_)), O_list).
assign(It, [], []).
assign(0, [A1|A],[Obj1|Objs]) :- K=random(k),
          A1=cluster(Obj1,K), assign(0, A, Objs).
assign(It, [A1|A],[Obj1|Objs]) :- It>0, closest(It-1, Obj1, Cluster),
           A1=cluster(Obj1,Cluster), assign(It, A, Objs).
```

The `objs_to_cluster` predicate defines the set of objects to be clustered. In the example instantiation given above, all `objs(X)` objects having some fixes defined were selected. For any iteration `It`, the `assign` predicate associates every object with its closest cluster centre, based on the results recursively obtained at the previous iteration, representing this information as terms of the form `cluster(object_ID, cluster_number)`. At iteration zero, the assignment object-cluster is random.

The above rules are based on the `closest` predicate, which we do not define here for lack of space, which finds, for each object `Obj1`, the cluster it should belong to. Such task requires other sub-predicates, which define how to compute both the cluster centres and the distance between objects. The computation of centres is accomplished by implementing a very natural and simple solution: a centre will be an object whose position at time t is the average of the positions of all objects in its cluster at time t.

The distance between objects can be defined in several ways, depending, e.g., on the meaning given to clusters or the coarseness allowed in the computation. One example of coarse but simple distance has been implicitly given in the previous section, where the similarity between two animals were defined as the percentage of mutually contemporary fixes (see the `couple` predicate, on page 6). In that case, only the explicit information on fixes has been exploited, not considering the whole trajectory followed by objects. A different and more precise solution, then, should take into account the position of objects for each time instant. Following this idea, a simple general approach to compute the distance $D(o_1, o_2)$ between two objects o_1 and o_2, whose positions along time $o_1(t)$ and $o_2(t)$ are defined over a time interval T, can be described by the following expression:

$$D(o_1, o_2) = \Phi(d_{o_1,o_2})\big|_T$$

where the first parameter of the schema, $d_{o_1,o_2}(t)$, is a distance measure between $o_1(t)$ and $o_2(t)$, and the second one, $\Phi(f)|_T$, is a functional computed over function f and domain T and returns a real value. In the STACLP rules defining the distance used in the example of this paper (not shown here for lack of space),

$d()$ is instantiated as the Euclidean distance on $\mathbf{R}^2$, and $\Phi()$ is the *average* functional, thus modelling $D(o_1, o_2)$ as the average Euclidean distance between o_1 and o_2. However, such parameters are modular components of the clustering algorithm, and therefore can be easily instantiated with other functions, as those described in [10] (e.g., other Minkowski's metrics for $d()$, and *min* or *max* functionals for $\Phi()$), thus defining new distance notions for $D(o_1, o_2)$. Thanks to the linear interpolation model adopted, the computation of such distance functions can be realized in linear time w.r.t. the number of fixes of each object.

Knowledge Discovery on Trajectories In this section we provide a very compact STACLP example program which shows how using the clustering tool can yield an alternative, more sophisticated, solution to the problem of discovering animal couples or herds.

In Section 4.1, a fully deductive approach has already been presented, where a simple criterion was adopted, based on contemporary fixes, to discover animal couples. A more precise and general solution to the problem can be achieved by simply noticing that animal couples and animal herds are simply groups of animal which, in general, move together. This can be straightforwardly rephrased saying that animal herds are clusters of animal individuals whose mutual distance is, on average, small. This leads to the following STACLP formalisation, where trajectories of animals are clustered using the k-means algorithm, and focusing on a time interval $[t_{start}, t_{end}]$:

```
objs_to_cluster(O_list) :-
              findall(fix(X) atp (_,_) at _, X=cut_obj(_), O_list).

fix(cut_obj(X)) atp (X,Y) at t_start :- traj(obj(X)) atp (X,Y) at t_start.
fix(cut_obj(X)) atp (X,Y) at t_end :- traj(obj(X)) atp (X,Y) at t_end.
fix(cut_obj(X)) atp (X,Y) at T :- T > t_start, T<t_end,
                                fix(obj(X)) atp (X,Y) at T.

cluster(K,O_list) :- findall( Obj,
                    (assignments(A), member(cluster(Obj,K), A)),
                    O_list).

couples([Obj1, Obj2]) :- cluster(_, [Obj1, Obj2]),
                        sex(Obj1, S1), sex(Obj2, S2), S1 != S2.

herds(O_list) :- cluster(_, O_list), length(O_list, N), N>=min_herd_size.
```

The first rule redefines the `objs_to_cluster` predicate in order to cluster the new `cut_obj(X)` objects, obtained by clipping the trajectories of the original `obj(X)` objects on the $[t_{start}, t_{end}]$ time interval (see the above definitions for `fix(cut_obj(X))`). In the rules defining couples and herds, we assumed (i) to be interested in couples of different sex, and that (ii) a necessary (and sufficient) condition for a group of animals to be a herd is that its size is not smaller that a given threshold. Of course, it is easy to insert more complex conditions on the properties of the group and of the animals it contains (e.g., checking the respect of given proportions in the number of male and female individuals).

5 Conclusions

The framework we presented provides the user with high level mechanisms to represent and reason on spatio-temporal data. The peculiarity of this approach is that it exhibits both deductive and inductive capabilities, thus offering the possibility to make analysis driven by domain expert rules (deduction) and driven by observations (induction). Furthermore we sketched how STACLP can be successfully applied to a concrete case study concerning behavioural ecology.

Currently we are improving the implementation of STACLP, which is at a prototype stage, and we intend to study how to introduce other knowledge discovery techniques, such as classification and frequent patterns.

Another promising direction we addressed concerns qualitative spatial reasoning. Some preliminary results are presented in [12] and we would like to analyse forms of qualitative reasoning on trajectories.

References

1. A.I. Abdelmoty, N.W. Paton, M.H. Williams, A.A.A. Fernandes, M.L. Barja, and A. Dinn. Geographic Data Handling in a Deductive Object-Oriented Database. In *DEXA Conf.*, volume 856 of *LNCS*, pages 445–454. Springer, 1994.
2. T. Abraham. *Knowledge Discovery in Spatio-Temporal Databases*. PhD thesis, School of Computer and Information Science, Faculty of Information Technology, University of South Australia, 1999.
3. T. Ceccarelli, D. Centeno, F. Giannotti, A. Massolo, C. Parent, A. Raffaetà, C. Renso, S. Spaccapietra, and F. Turini. The behaviour of the Crested Porcupine: the complete case study. Technical report, DeduGIS - EU WG, 2001.
4. J. Chomicki and P.Z. Revesz. Constraint-Based Interoperability of Spatiotemporal Databases. *GeoInformatica*, 3(3):211–243, 1999.
5. C. Faloutsos and K.-I. Lin. Fastmap: a fast algorithm for indexing of traditional and multimedia databases. In *SIGMOD Conf.*, pages 163–174. ACM, 1995.
6. T. Frühwirth. Temporal Annotated Constraint Logic Programming. *Journal of Symbolic Computation*, 22:555–583, 1996.
7. S. Gaffney and P. Smyth. Trajectory clustering with mixture of regression models. In *KDD Conf.*, pages 63–72. ACM, 1999.
8. S. Grumbach, P. Rigaux, and L. Segoufin. Spatio-Temporal Data Handling with Constraints. *GeoInformatica*, 5(1):95–115, 2001.
9. A. Ketterlin. Clustering sequences of complex objects. In *KDD Conf.*, pages 215–218. ACM, 1997.
10. M. Nanni. *Clustering Methods for Spatio-Temporal Data*. PhD thesis, Dipartimento di Informatica, Università di Pisa, 2002.
11. A. Raffaetà and T. Frühwirth. Spatio-Temporal Annotated Constraint Logic Programming. In *PADL'01*, volume 1990 of *LNCS*, pages 259–273. Springer, 2001.
12. A. Raffaetà, C. Renso, and F. Turini. Qualitative Spatial Reasoning in a Logical Framework. In *AI*IA Conf.*, volume 2829 of *LNAI*, pages 78–90, 2003.
13. S. Spaccapietra, editor. *Spatio-Temporal Data Models & Languages (DEXA Workshop)*. IEEE Computer Society Press, 1999.
14. M. F. Worboys. *GIS - A Computing Perspective*. Taylor & Francis, 1995.