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. Inductive methods can be exploited for data anlysis since they are capable to extract implicit knowledge from raw observations. However, extracted patterns are very seldom geographic knowledge pr?t-a-porter: it is necessary to reason on patterns and on pertinent background knowledge, evaluate patterns interestingness, refer them to geographic information.

The first step for allowing one to make a profitable use of data and extracted patterns 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. The approach we propose here is a first step to build a framework where a representational and a query language for spatio-temporal data mining are conciliated providing a uniform language that merges raw data representation, methods for

patterns extraction and reasoning formalisms over background knowledge, the integration of possibly other related georeferenced data and extracted patterns.

In order to support both deductive and inductive reasoning on spatio-temporal data we propose the formalism STACLP (Spatio-Temporal Annotated Constraint Logic Programming) [40] 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 [55, 49, 1, 19].

Information induction over spatio-temporal data is far from being a mature field, yet. In recent years, many algorithms and applications have been investigated in literature, which deal with spatial data (e.g., [12, 22, 37, 30]) *or* temporal data (e.g., [50, 18, 27]), either extending ideas coming from relational data mining or introducing new concepts of patterns and new computational approaches. On the contrary, only a limited number of proposals are actually available on mining methods which exploit both the spatial and the temporal components of data. In Section 2 a brief review is provided.

Section 3 introduces the language STACLP and Section 4 focuses on the representation of one of the most fundamental spatio-temporal objects, i.e. the trajectory. In Section 5 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 6 draws some conclusions and outlines future research directions.

2 Related Work

While a lot of effort has been spent in developing extensions of logic programming languages capable to manage time [38], the logic based languages for the handling of spatial information only deal with the qualitative representation and reasoning about space (see e.g. [43]). And also the few attempts to manipulate time and space have led to languages for qualitative spatio-temporal representation and reasoning [54]. On the other hand temporal [52, 13] and spatial [21, 39] database technologies are relatively mature, although, also in the database area, their combination is far from straightforward [6].

Our spatio-temporal language is close to the approaches based on constraint databases [5, 8, 19]. ¿From a database point of view, logic programs can represent deductive databases, i.e. relational databases enriched with intensional rules, constraint logic programs can represent constraint databases [26], and thus STACLP can represent spatio-temporal constraint databases. The spatio-temporal proposals in [5, 19] are extensions of languages originally developed to express

only spatial data. Thus the high-level mechanisms they offer are more oriented to query spatial data than temporal information. In fact, they can model only definite temporal information and there is no support for periodic, indefinite temporal data. On the contrary STACLP provides several facilities to reason on temporal data and to establish spatio-temporal correlations. For instance, it allows one to describe *continuous* change in time as well as [8] does, whereas both [5] and [19] can represent only *discrete* changes. Also indefinite spatial and temporal information can be expressed in STACLP, a feature supported only by the approach in [32].

Spatio-temporal data mining is a subfield of data mining and knowledge discovery, aimed at the extraction of spatial and temporal patterns and relationships not explicitly contained in the database. Introducing a spatial and/or temporal component to data has two main effects: on one hand, the complexity of the data mining task is highly increased, requiring to adopt suitable measures to contain the computation time; on the other hand, space and time are not simple attributes, since they have a specific semantics, and then new, ad hoc analysis tools should be developed to take full advantage of such information.

In the last ten years, several mining algorithms for temporal data have been presented in literature. Among the mainstream research subfields, we mention the mining of frequent patterns in transactional, timestamped databases such as sequential patterns [3, 50] and episodes [34], and the large area of time series mining: time series classification [18, 9, 10], sequential association rules [11, 23, 25], clustering [16, 20, 27] and anomaly detection [56].

Several approaches have also been proposed for mining spatial data (see [35], [48], [46] for reviews of recent results): spatial trend detection [12], clustering [22, 44, 4, 24, 57], outlier detection [37, 47], association/co-location rules [30, 45] and classification [29, 12, 31].

In the context of spatio-temporal data minig, where the spatial and temporal component are expected to be used together, [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, only a limited number of concrete proposals are available for the mining of spatio-temporal data, such as sequential patterns [53], movement prediction [51] and clustering [36, 28, 17, 14]. In this work, in particular, a clustering algorithm for trajectories is designed within the STACLP system, following the solution described in [36]. However a few alternative approaches can be found in literature: [28] considers generic sequences together with a conceptual hierarchy over the sequence elements, used to compute both the cluster representatives and the distance between two sequences. In [17], a model-based clustering method for continuous trajectories is proposed, which puts together objects which can be obtained from a common core trajectory by adding noise with normal distribution. Finally, [14] proposes a general mapping from any data space to an Euclidean space, where any standard clustering algorithm can be applied.

3 STACLP: a Spatio-Temporal Language

In this section we present the language STACLP [40], which extends Temporal Annotated Constraint Logic Programming [15], 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.

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:

- $\texttt{at}\,T$ and $\texttt{atp}\,(X, Y)$ are used to express that a formula holds in a time or spatial point, respectively.
- $\texttt{th}\,I$, $\texttt{thr}\,R$ are used to express that a formula holds *throughout*, i.e., at *every* point, in the temporal interval or the spatial region, respectively.
- $\texttt{in}\,I$, $\texttt{inr}\,R$ 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 $\sqsubseteq$. 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, $A\,\beta \Rightarrow A\,\alpha$. This partial order is used in the definition of new inference rules. In addition to Modus Ponens, STACLP has the two inference rules below:

$$\frac{A\,\alpha \qquad \gamma \sqsubseteq \alpha}{A\,\gamma} \quad \textit{rule } (\sqsubseteq) \qquad\qquad \frac{A\,\alpha \qquad A\,\beta \qquad \gamma = \alpha \sqcup \beta}{A\,\gamma} \quad \textit{rule } (\sqcup)$$

The rule $(\sqsubseteq)$ states that if a formula holds with some annotation, then it also holds with all annotations that are smaller according to the partial ordering.

The rule ($\sqcup$) says that if a formula holds with some annotation α and the same formula holds with another annotation β then it holds with the least upper bound $\alpha \sqcup \beta$ of the two annotations.

Next, we introduce the *constraint theory for temporal and spatial annotations*. A constraint theory is a non-empty, consistent first order theory that axiomatises the meaning of the constraints. Besides an axiomatisation of the total order relation $\leq$ on the set of points, the constraint theory includes the axioms in Table 1 defining the partial order on temporal and spatial annotations. The first two axioms state that $\mathtt{th}\, I$ and $\mathtt{in}\, I$ are equivalent to $\mathtt{at}\, t$ when the time period I consists of a single time point t. Next, if a formula holds at every point of a time period, then it holds at every point in all sub-periods of that period (($\mathtt{th}\ \sqsubseteq$) axiom). On the other hand, if a formula holds at some points of a time period then it holds at some points in all periods that include this period (($\mathtt{in}\ \sqsubseteq$) axiom). The axioms for spatial annotations are analogously defined.

($\mathtt{at}\,\mathtt{th}$)	$\mathtt{at}\, t = \mathtt{th}\,[t,t]$
($\mathtt{at}\,\mathtt{in}$)	$\mathtt{at}\, t = \mathtt{in}\,[t,t]$
($\mathtt{th}\ \sqsubseteq$)	$\mathtt{th}\,[s_1,s_2] \sqsubseteq \mathtt{th}\,[r_1,r_2] \Leftrightarrow r_1 \leq s_1,\ s_2 \leq r_2$
($\mathtt{in}\ \sqsubseteq$)	$\mathtt{in}\,[r_1,r_2] \sqsubseteq \mathtt{in}\,[s_1,s_2] \Leftrightarrow r_1 \leq s_1,\ s_2 \leq r_2$
($\mathtt{atp}\,\mathtt{thr}$)	$\mathtt{atp}\,(x,y) = \mathtt{thr}\,[(x,x),(y,y)]$
($\mathtt{atp}\,\mathtt{inr}$)	$\mathtt{atp}\,(x,y) = \mathtt{inr}\,[(x,x),(y,y)]$
($\mathtt{thr}\ \sqsubseteq$)	$\mathtt{thr}\,[(x_1,x_2),(y_1,y_2)] \sqsubseteq \mathtt{thr}\,[(x_1',x_2'),(y_1',y_2')] \Leftrightarrow$ $x_1' \leq x_1,\ x_2 \leq x_2', y_1' \leq y_1,\ y_2 \leq y_2'$
($\mathtt{inr}\ \sqsubseteq$)	$\mathtt{inr}\,[(x_1',x_2'),(y_1',y_2')] \sqsubseteq \mathtt{inr}\,[(x_1,x_2),(y_1,y_2)] \Leftrightarrow$ $x_1' \leq x_1,\ x_2 \leq x_2', y_1' \leq y_1,\ y_2 \leq y_2'$

Table 1. Axioms for the partial order on annotations.

3.1 Combining Spatial and Temporal Annotations

In order to obtain spatio-temporal annotations the spatial and temporal annotations are combined by considering pairs of annotations as a new class of annotations. Let us first introduce the general idea of pairing of annotations.

Definition 1. *Let* $(A, \sqsubseteq_A)$ *and* $(B, \sqsubseteq_B)$ *be two disjoint classes of annotations with their partial order. Their* pairing *is the class of annotations* $(A * B, \sqsubseteq_{A*B})$ *defined as* $A * B = \{\alpha\beta, \beta\alpha \mid\ \alpha \in A,\ \beta \in B\}$ *and* $\gamma_1 \sqsubseteq_{A*B} \gamma_2$ *whenever*

$$((\gamma_1 = \alpha_1\beta_1 \wedge \gamma_2 = \alpha_2\beta_2) \vee (\gamma_1 = \beta_1\alpha_1 \wedge \gamma_2 = \beta_2\alpha_2)) \ \wedge (\alpha_1 \sqsubseteq_A \alpha_2 \ \wedge\ \beta_1 \sqsubseteq_B \beta_2)$$

In our case the spatio-temporal annotations are obtained by considering the pairing of spatial and temporal annotations.

Definition 2 (Spatio-temporal annotations). *The class of spatio-temporal annotations is the pairing of the spatial annotations* Spat *built from* `atp`, `thr` *and* `inr` *and of the temporal annotations* Temp*, built from* `at`, `th` *and* `in`*, i.e.* Spat$*$Temp.

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.

4 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 [8]).

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 $t \in [t_1, t_2]$:

$$\begin{aligned}(x - x_1)(t_2 - t_1) \ - \ (x_2 - x_1)(t - t_1) \ = \ 0, \ \textit{and} \\ (y - y_1)(t_2 - t_1) \ - \ (y_2 - y_1)(t - t_1) \ = \ 0\end{aligned}$$

where (x_1, y_1, t_1) and (x_2, y_2, t_2) are two consecutive location points (i.e., there are not other locations in the time interval $[t_1, t_2]$).

Specifically, the N locations 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 locations 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, Y) 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 location points, ensures that no other observation exists between times t_1 and t_2, i.e., the interpolation is performed only between consecutive location points.

5 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 locations, 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 location. 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 [7], 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 5.1 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 5.2, instead, is devoted to introduce mechanisms to support inductive analysis, describing how these tools can be successfully used in our case study.

5.1 Deductive Analysis in STACLP

To model the spatio-temporal locations 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 location 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 locations 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.

Finally, we show the STACLP code to compute a representation of animal home ranges which can enable an effective detection of their changes, following the ideas presented in Sect. **??**.

We recall that we want to compute the home range of an animal within a given time interval `[t_start,t_end]` in a "continuous" way. Concretely, home ranges are computed on overlapping time intervals, each of duration `Wt` and each shifted of δ`t` w.r.t. the previous one. We assume that `Wt <= t_end - t_start`, i.e., at least one of such intervals can fit in `[t_start,t_end]`.

```
homerange(Fix_list, Home) :- <ad hoc query/external call>

home(Id, Home) at t_start:- findall((T,X,Y),
    (fix(Id) atp (X,Y) at T, T>= t_start, T<t_start+Wt), Fix_list),
    homerange(Fix_list, Home).
home(Id, Home) at T:- home(Id, _) at T_prev,
```

```
T=T_prev+δt, T + Wt <= t_end,
findall((T_fix,X,Y),
        (fix(Id) atp (X,Y) at T_fix, T_fix >= T, T_fix < T+Wt),
        Fix_list),
homerange(Fix_list, Home).
```

Given a list of fixes, the predicate `homerange` returns the corresponding home range `Home` by simply calling a routine provided by an external application. More details on the use of built-in predicates to directly invoke external functions can be found in [33, 41].

The predicate `home` returns the home range `Home` for an animal `Id` at regular time points, i.e. at `t_start` and at time points shifted from `t_start` of a multiple of δ`t`. The first rule states that the home range for animal `Id` at the time instant `t_start` is obtained by finding all the fixes included in the interval `[t_start, t_start+Wt]` and then applying the home range routine to these fixes. The second rule manages the remaining time points and is defined in a similar way: it computes the home range using the fixes within the interval `[T, T+Wt]`, provided that `T` is shifted from `t_start` by a multiple of δ`t` and `T+Wt <= t_end`.

5.2 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 to 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)`".

```
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(_)` 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.

```
closest(Iter, Obj, Cluster) :- best_dist(Iter, Obj, k, Cluster, D).
best_dist(Iter, Obj, 1, 1, D) :- distance(centre(Iter, 1), Obj, D).
best_dist(Iter, Obj, K, Cluster, D) :- K>1,
             distance(centre(Iter, K), Obj, D1),
             best_dist(Iter, Obj, K-1, Cluster2, D2),
             if D1 < D2 then Cluster=K, D=D1
                        else Cluster=Cluster2, D=D2.
```

Here the selection of the closest cluster centre is implemented, simply scanning all the `k` centres obtained for the previous iteration, searching for the minimum value of the distance w.r.t. the object to assign. Here it appears the `distance` predicate, defined later in this section. In the following last set of rules, the centre of each cluster for any iteration is defined by setting its coordinates to the average values taken by all objects in the cluster (notice that a

predicate `sum_pairs` is used to sum the single components of couples: since it is quite trivial to implement, its definition is omitted).

```
fix(centre(Iter,K)) atp (X,Y) at T :-
    objs_to_cluster(O_list), assign(Iter, A, O_list),
    member(cluster(Obj,K), A), fix(Obj) atp (_, _) at T,
    compute_avg_position(A, K, T, X, Y).

compute_avg_position(A, K, T, X, Y) :-
      findall((X1, Y1),
              (member(cluster(O,K),A), traj(O) atp (X1, Y1) at T), L),
      sum_pairs(L, (Xsum, Ysum)), length(L,N), N>0,
      X=Xsum/N, Y=Ysum/N.

fix(centre(K)) atp (X,Y) at T :-
               fix(centre(max_n_iters, K)) atp (X,Y) at T.

assignments(A) :- objs_to_cluster(O_list),
                  assign(max_n_iters, A, O_list).
```

Notice that in the definition of `compute_avg_position` the `traj` predicate is used to interpolate the position of objects, since the fixes of an object could be not aligned to the fixes of the others. The final result of the clustering process is represented by the cluster assignments and the means of the centres obtained when the maximum number of iterations has been reached – in particular, such centres will coincide with a local minimum of the clustering process if the algorithm converges in less than `max_n_iters` iterations.

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 in Section 5.1). 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 given below, $d()$ 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 [36] (e.g., other Minkowski's metrics for $d()$, and *min* or *max* functionals for $\Phi()$), thus defining new distance notions for $D(o_1, o_2)$.

Computing the average Euclidean distance between moving objects requires to calculate an integral of the Euclidean distance formula over a given time interval $[t_{start}, t_{end}]$. Thanks to the linear interpolation model adopted, such computation can be realised in linear time w.r.t. the number of fixes of each object [36]. This is due to the fact that the integration interval can be broken down to subintervals and in each of them the integral can be symbolically solved and thus computed in constant time. The following rules essentially find such subintervals, use a predicate `compute_sub` (not described here, as well as `sort_without_duplicates`, for sake of brevity) to compute local integrals, and aggregate them.

```
distance(O1,O2,D) :- collect_fixes(O1,O2,Fixes),
                     integral(O1,O2,Fixes,Int), D=I/(t_end-t_start).

collect_fixes(O1,O2,Fixes) :-
                    findall(T, fix(O1) atp (_,_) at T, L1),
                    findall(T, fix(O2) atp (_,_) at T, L2),
                    append(L1,L2,L).
                    sort_without_duplicates(L, Fixes).

integral(O1,O2,[_],0).
integral(O1,O2,[T1|[T2|T]], Int) :-
      traj(O1) atp (X11,Y11) at T1, traj(O1) atp (X12,Y12) at T2,
      traj(O2) atp (X21,Y21) at T1, traj(O2) atp (X22,Y22) at T2,
      compute(X11, Y11, X12, Y12, X21, Y21, X22, Y22, T1, T2, Int1),
      integral(O1,O2,[T2|T], Int2), Int = Int1 + Int2.
```

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 5.1, 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(X, (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_member(K,O_list) :- findall( Obj,
                            (assignments(A), member(cluster(Obj,K), A)),
```

```
                          O_list).
couples([Obj1, Obj2]) :- cluster_member(_, [Obj1, Obj2]),
                         sex(Obj1, S1), sex(Obj2, S2), S1 != S2.
herds(O_list) :- cluster_member(_, 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))`). The `cluster_member` predicate provides the list of objects belonging to a cluster and it is exploited to find out couples and herds by checking the size of clusters. In the rules defining couples and herds, we assumed (i) to be interested in clusters of two individuals 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).

Another aspect of interest for the analyst is the existence of animals that, in reaction to an event, leave the herd. This can be seen as an alternative measure of dispersion of the herd, based on the behaviour of single animals w.r.t. the herd they belong to instead of considering only the overall behaviour of the heard.

Animals which abandon their herd can be defined as individuals that before the event belong to a given cluster, but, after the event, they *move* closer to other clusters. We therefore need to cluster animals w.r.t. their trajectory before the event, and then check if the cluster assignments are still valid also after event. The first step has already been performed in the previous example. To implement the second one, we compute first the centre of each cluster after the event (essentially with the same rule used to define the clustering engine), then for each object we find the closest centre to verify if it corresponds to the cluster the object belongs to:

```
fix(centre(after_event,K)) atp (X,Y) at T :-
    cluster(K, O_list), rename_objs(K, O_list, A_list),
    member(Obj, O_list), fix(Obj) atp (_,_) at T,
    compute_avg_position(A_list, K, T, X, Y).
rename_objs(K, [], []).
rename_objs(K, [before(X) | Objs], [A1|As]) :- A1=cluster(after(X),K),
             rename_objs(K, Objs, As).
run_away(obj(X)) :- cluster(K_before, O_list),
             member(before(X), O_list),
             closest(after_event, after(X), K_after),
             K_after != K_before.
```

The centre of each cluster k is named `centre(after_event, k)`, in order to obtain object names syntactically compatible with the rules which define the clustering engine, and in particular those which define the `closest`, used to

define the `run_away` predicate shown above. Finally, the `rename_objs` predicate has the simple purpose of converting the cluster assignment of each `before(X)` object into the equivalent assignment for the corresponding `after(X)` object (i.e., the second half of each trajectory is labelled according to the clustering obtained on the first half).

6 Conclusions

The main aim of the framework we presented is to provide 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 both exploiting domain expert rules and general background knowledge (deduction) and driven by observations (induction). Furthermore we sketched how this approach can be successfully applied to a concrete case study concerning behavioural ecology that well represents this two kinds of reasoning.

We are currently improving the implementation of STACLP, which is at a prototype stage and lacks of optimization techniques. As a future research direction we are moving towards the introduction of other knowledge discovery techniques, such as classification and frequent patterns, in this framework. This leads to challenging and interesting research problems as well a wide range of possible applications related to mobile devices. As an example, classification technique applied to trajectories can be exploited to predict the future direction of a moving objects. For example detecting frequent patterns of a number of trajectories representing car movements can identify routes with high traffic density depending from the time of the day.

Another promising direction we intend to address concerns qualitative spatio-temporal reasoning. Starting from some preliminary results presented in [42], we aim at defining forms of qualitative reasoning on trajectories thus providing support for *qualitative spatio-temporal reasoning*, possibly enriched with uncertainty information. As an example, a typical qualitative spatio temporal query can be to find out whether, when and with which degree of uncertainty a given trajectory crosses a specific area (e.g. traffic related to a particular event in a given area).

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. R. Agrawal and R. Srikant. Mining sequential patterns. In Philip S. Yu and Arbee S. P. Chen, editors, *Eleventh International Conference on Data Engineering*, pages 3–14, Taipei, Taiwan, 1995. IEEE Computer Society Press.

4. M. Ankerst, M. Breunig, H.-P. Kriegel, and J. Sander. Optics: Ordering points to identify the clustering structure. In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'99)*. ACM Press, 1999.
5. A. Belussi, E. Bertino, and B. Catania. An extended algebra for constraint databases. *IEEE TKDE*, 10(5):686–705, 1998.
6. M.H. Böhlen, C.S. Jensen, and M.O. Scholl, editors. *Spatio-Temporal Database Management*, volume 1678 of *Lecture Notes in Computer Science*. Springer, 1999.
7. 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.
8. J. Chomicki and P.Z. Revesz. Constraint-Based Interoperability of Spatiotemporal Databases. *GeoInformatica*, 3(3):211–243, 1999.
9. P. Cotofrei. Statistical temporal rules. In *Proceedings of the 15th Conf. on Computational Statistics*, 2002.
10. P. Cotofrei and K. Stoffel. Classification rules + time = temporal rules. In *Proceedings of the 2002 Int. Conf. on Computational Science*, 2002.
11. G. Das, K.-I. Lin, H. Mannila, G. Renganathan, and P. Smyth. Rule discovery from time series. In *Knowledge Discovery and Data Mining*, pages 16–22, 1998.
12. M. Ester, H.-P. Kriegel, and J. Sanders. Algorithms and applications for spatial data mining. In *[35]*, pages 160–187.
13. O. Etzion, S. Jajodia, and S. Sripada, editors. *Temporal Databases: Research and Practice*, volume 1399 of *Lecture Notes in Computer Science*. Springer, 1998.
14. 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.
15. T. Frühwirth. Temporal Annotated Constraint Logic Programming. *Journal of Symbolic Computation*, 22:555–583, 1996.
16. T. C. Fu, F. L. Chung, V. Ng, and R. Luk. Pattern discovery from stock time series using self-organizing maps. In *Workshop Notes of KDD2001 Workshop on Temporal Data Mining*, 2001.
17. S. Gaffney and P. Smyth. Trajectory clustering with mixture of regression models. In *KDD Conf.*, pages 63–72. ACM, 1999.
18. P. Geurts. Pattern extraction for time series classification. *Lecture Notes in Computer Science*, 2168:115–127, 2001.
19. S. Grumbach, P. Rigaux, and L. Segoufin. Spatio-Temporal Data Handling with Constraints. *GeoInformatica*, 5(1):95–115, 2001.
20. S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams. In *IEEE Symposium on Foundations of Computer Science*, pages 359–366, 2000.
21. R.H. Güting. An Introduction to Spatial Database Systems. *VLDB Journal*, 3(4):357–400, 1994.
22. J. Han, M. Kamber, and A. K. H. Tung. Spatial clustering methods in data mining: a survey. In *[35]*, pages 188–217.
23. S. K. Harms, J. Deogun, and T. Tadesse. Discovering sequential association rules with constraints and time lags in multiple sequences. In *Proceedings of the 13th Int. Symposium on Methodologies for Intelligent Systems*, pages 432–441, 2002.
24. A. K. H. Tung J. Hou and J. Han. Spatial clustering in the presence of obstacles. In *Proceedings of the 17th International Conference on Data Engineering*, 2001.
25. X. Jin, L. Wang, Y. Lu, and C. Shi. Indexing and mining of the local patterns in sequence database. In *Proceedings of the Third International Conference on Intelligent Data Engineering and Automated Learning*, pages 68–73. Springer-Verlag, 2002.

26. P.C. Kanellakis, G.M. Kuper, and P.Z. Revesz. Constraint query languages. *Journal of Computer and System Sciences*, 51(1):26–52, 1995.
27. E. Keogh, J. Lin, , and W. Truppel. Clustering of time series subsequences is meaningless: Implications for past and future research. In *Proceedings of the 3rd IEEE International Conference on Data Mining*, 2003.
28. A. Ketterlin. Clustering sequences of complex objects. In *KDD Conf.*, pages 215–218. ACM, 1997.
29. K. Koperski. *A Progressive Refinement Approach to Spatial Data Mining*. PhD thesis, Simon Frasery University, 1999.
30. K. Koperski and J. Han. Discovery of spatial association rules in geographic information databases. In *Advances in Spatial Databases, Proc. of 4th Symp. SSD'95*, pages 47–66, Berlin, 1995. Springer-Verlag.
31. K. Koperski, J. Han, and N. Stefanovic. An efficient two-step method for classification of spatial data, 1998.
32. M. Koubarakis and S. Skiadopoulos. Tractable Query Answering in Indefinite Constraint Databases: Basic Results and Applications to Querying Spatiotemporal Information. In *[6]*, pages 204–223, 1999.
33. P. Mancarella, A. Raffaetà, C. Renso, and F. Turini. Integrating Knowledge Representation and Reasoning in Geographical Information Systems. *International Journal of GIS*. To appear.
34. H. Mannila, H. Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997.
35. H. J. Miller and J. Han, editors. *Geographic Data Mining and knowledge Discovery*. Taylor & Francis, 2001.
36. M. Nanni. *Clustering Methods for Spatio-Temporal Data*. PhD thesis, Dipartimento di Informatica, Università di Pisa, 2002.
37. R. T. Ng. Detecting outliers from large datasets. In *[35]*, pages 218–235.
38. M. A. Orgun and W. Ma. An Overview of Temporal and Modal Logic Programming. In *ICTL'94*, volume 827 of *Lecture Notes in Artificial Intelligence*, pages 445–479. Springer, 1994.
39. J. Paredaens. Spatial databases, the final frontier. In *ICDT'95*, volume 893 of *Lecture Notes in Computer Science*, pages 14–32. Springer, 1995.
40. A. Raffaetà and T. Frühwirth. Spatio-Temporal Annotated Constraint Logic Programming. In *PADL'01*, volume 1990 of *LNCS*, pages 259–273. Springer, 2001.
41. A. Raffaetà, C. Renso, and F. Turini. Enhancing GISs for Spatio-Temporal Reasoning. In *GIS'02*, pages 35–41. ACM, 2002.
42. 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.
43. D. Randell, Z. Cui, and A. Cohn. A Spatial Logic based on Regions and Connection. In *KR1992*, pages 165–176. Morgan Kaufmann, 1992.
44. G. Sheikholeslami, S. Chatterjee, and A. Zhang. Wavecluster: A multi-resolution clustering approach for very large spatial databases. In *Proceedings of the 24rd International Conference on Very Large Data Bases*, pages 428–439. Morgan Kaufmann Publishers Inc., 1998.
45. S. Shekhar and Y. Huang. Discovering spatial co-location patterns: A summary of results. In Lecture Notes on Computer Science, editor, *SSTD*, volume 2121, pages 236–256, 2001.
46. S. Shekhar, Y. W. Huang, C. T. Lu S., and Chawla. *Data Mining for Scientific and Engineering Applications*, chapter What's spatial about spatial data mining: three case studies. Kluwer Academic Publishers, 2001.

47. S. Shekhar, C.-T. Lu, and P. Zhang. Detecting graph-based spatial outliers: algorithms and applications (a summary of results). In *Proceedings of the 7th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 371–376. ACM Press, 2001.
48. S. Shekhar, P. Zhang, R. R. Vatsavai, and Y. Huang. Research accomplishments and issues on spatial data mining. In *White paper of the Geospatial Visualization and Knowledge Discovery Workshop*, Lansdowne, Virginia, 2003. http://www.ucgis.org/Visualization/.
49. S. Spaccapietra, editor. *Spatio-Temporal Data Models & Languages (DEXA Workshop)*. IEEE Computer Society Press, 1999.
50. R. Srikant and R. Agrawal. Mining sequential patterns: generalisations and performance improvements. In *Proceedings of the 5th Int. Conf. on Extending Database Technology (EDBT'96)*, pages 3–17, 1996.
51. N. Sumpter and A. Bulpitt. Learning spatio-temporal patterns for predicting object behaviour. *Image and Vision Computing*, 18(9):697–704, 2000.
52. A. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. Snodgrass editors. *Temporal Databases: Theory, Design, and Implementation.* Benjamin/Cummings, 1993.
53. I. Tsoukatos and D. Gunopulos. Efficient mining of spatiotemporal patterns. In Lecture Notes on Computer Science, editor, *Proceedings of SSTD 2001*, volume 2121, pages 425–442, 2001.
54. F. Wolter and M. Zakharyaschev. Spatio-temporal representation and reasoning based on RCC-8. In *KR2000*, pages 3–14. Morgan Kaufmann, 2000.
55. M. F. Worboys. *GIS - A Computing Perspective.* Taylor & Francis, 1995.
56. T. Yairi, Y. Kato, and K. Hori. Fault detection by mining association rules from house-keeping data. In *Proc. of International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2001.
57. T. Zhang, R. Ramakrishnan, and M. Livny. Birch: an efficient data clustering method for very large databases. In *Proceedings of the 1996 ACM SIGMOD international conference on Management of data*, pages 103–114. ACM Press, 1996.