A Declarative Framework for 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

Summary. We present a framework for a declarative approach to spatio-temporal reasoning on geographical data. We introduce a constraint logical language STACLP, providing a set of spatial and temporal primitive operators that allow the user to perform temporal reasoning on spatial data. Formulae can be annotated with labels (annotations) to represent both temporal and spatial information, and relations between these labels can be expressed by using constraints. The role of such formalism can be manifold: first, it can be used as an advanced spatio-temporal query language on geographical data. Second, it can be exploited as a deductive rule-based approach to represent domain knowledge on such data. Third, it is well suited to represent trajectories of moving objects. Such trajectories can be analysed by using inductive techniques, like clustering, in order to find common movement patterns. It is worth to point out that STACLP allows one to tackle several analysis tasks requiring the integration of deductive and inductive capabilities. This is exemplified by means of a case study in the field of behavioural ecology.

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) [20], based on constraint logic programming extended with annotations. The basic 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. The computation of clusters of trajectories is the case we discuss in detail in the chapter. However we believe that it is paradigmatic for the general approach to the definition of knowledge discovery methods in STACLP.

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 [25, 23]. For instance [1] expresses spatial data in an object-oriented paradigm and builds a deductive part to infer knowledge from the spatial objects. The system DEDALE [12, 13], instead, relies on a logical model based on linear constraints, which generalises the constraint database model of [16]. In DEDALE both time and space are modelled in a uniform way by using constraints (for more details see Sect. 4). Finally, we recall MuTACLP [21, 18], a constraint logic based knowledge representation language that offers facilities for handling spatio-temporal information and some basic operators for combining different spatio-temporal knowledge bases. In MuTACLP temporal information is represented by annotations whereas spatial information is encoded into the formulae, by using constraints. This leads to a mismatch of conceptual levels and a loss of simplicity. In STACLP we overcome this mismatch, by defining a uniform setting where spatial information is represented by means of annotations, so that the advantages of using annotations apply to the spatial dimension as well.

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. In addition to the trivial extensions of spatial clustering methods, which treat time as just another generic dimension, three main approaches can be found in literature, focusing on the clustering of moving objects: [17] 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 [11], 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, [9] proposes a general mapping from any data space to an Euclidean space, where any standard clustering algorithm can be applied.

In Sects. 2 and 3 we introduce syntax and semantics of STACLP along with some simple examples, that illustrate its knowledge representation capabilities. In Sect. 4 we deal with the representation of one of the most fundamental spatio-temporal objects, i.e. the trajectory. In Sect. 5 we set the stage for the rest of the chapter, by introducing the requirements of a complex application dealing with the analysis of the behaviour of a nice group of animals, crested porcupines, living in a natural park of Tuscany (Italy). In Sect. 6 we present how STACLP allows us to solve part of the above problems by using the deductive capabilities of the language, while Sect. 7 deals with implementing a clustering method in STACLP, how to specialise it for trajectories, and how to solve the problems of the application that require such a kind of inductive reasoning. Finally, Sect. 8 draws some conclusions and outlines future research directions.

2 STACLP: a Spatio-Temporal Language

In this section we present the language STACLP [20], which extends Temporal Annotated Constraint Logic Programming [10], 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.

2.1 Time and Space

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.

2.2 Annotations and Annotated Formulae

An *annotated formula* is 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.

$\mathtt{th}\,I$, $\mathtt{thr}\,R$ 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), 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:

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

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 subperiods of that period ((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 ((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

2.3 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. *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.

To clarify the meaning of our spatio-temporal annotations, we present some examples of their formal definition in terms of `at` and `atp`. Let t be a time point, $J = [t_1, t_2]$ be a time period, $s = (x, y)$ be a spatial point and $R = [(x_1, x_2), (y_1, y_2)]$ be a rectangle.

- The equivalent annotated formulae $A\,\mathtt{atp}\,s\,\mathtt{at}\,t$ and $A\,\mathtt{at}\,t\,\mathtt{atp}\,s$ mean that A holds at time point t in the spatial point s.
- The annotated formula $A\,\mathtt{thr}\,R\,\mathtt{th}\,J$ means that A holds *throughout* the time period J and at every spatial point in R. The definition of such a formula in terms of `atp` and `at` is:

$$A\,\mathtt{thr}\,R\,\mathtt{th}\,J \ \Leftrightarrow\ \forall t \in J.\ \forall s \in R.\ A\,\mathtt{atp}\,s\,\mathtt{at}\,t.$$

The formula $A\,\mathtt{th}\,J\,\mathtt{thr}\,R$ is equivalent to the formula above because one can be obtained from the other just by swapping the universal quantifiers.

(1)	$\mathbf{thr}\,[(x_1,x_2),(y_1,y_2)]\mathbf{th}\,[t_1,t_2] \sqcup \mathbf{thr}\,[(x_1,x_2),(z_1,z_2)]\mathbf{th}\,[t_1,t_2] = \mathbf{thr}\,[(x_1,x_2),(y_1,z_2)]\mathbf{th}\,[t_1,t_2] \Leftrightarrow y_1 \leq z_1, z_1 \leq y_2, y_2 \leq z_2$
(1′)	axiom obtained by swapping the annotations in (1).
(2)	$\mathbf{thr}\,[(x_1,x_2),(y_1,y_2)]\mathbf{th}\,[t_1,t_2] \sqcup \mathbf{thr}\,[(z_1,z_2),(y_1,y_2)]\mathbf{th}\,[t_1,t_2] = \mathbf{thr}\,[(x_1,z_2),(y_1,y_2)]\mathbf{th}\,[t_1,t_2] \Leftrightarrow x_1 \leq z_1, z_1 \leq x_2, x_2 \leq z_2$
(2′)	axiom obtained by swapping the annotations in (2).
(3)	$\mathbf{thr}\,[(x_1,x_2),(y_1,y_2)]\mathbf{th}\,[s_1,s_2] \sqcup \mathbf{thr}\,[(x_1,x_2),(y_1,y_2)]\mathbf{th}\,[r_1,r_2] = \mathbf{thr}\,[(x_1,x_2),(y_1,y_2)]\mathbf{th}\,[s_1,r_2] \Leftrightarrow s_1 \leq r_1, r_1 \leq s_2, s_2 \leq r_2$
(3′)	axiom obtained by swapping the annotations in (3).
(4)	$\mathbf{inr}\,[(x_1,x_2),(y_1,y_2)]\mathbf{th}\,[s_1,s_2] \sqcup \mathbf{inr}\,[(x_1,x_2),(y_1,y_2)]\mathbf{th}\,[r_1,r_2] = \mathbf{inr}\,[(x_1,x_2),(y_1,y_2)]\mathbf{th}\,[s_1,r_2] \Leftrightarrow s_1 \leq r_1, r_1 \leq s_2, s_2 \leq r_2$
(5)	$\mathbf{in}\,[t_1,t_2]\mathbf{thr}\,[(x_1,x_2),(y_1,y_2)] \sqcup \mathbf{in}\,[t_1,t_2]\mathbf{thr}\,[(x_1,x_2),(z_1,z_2)] = \mathbf{in}\,[t_1,t_2]\mathbf{thr}\,[(x_1,x_2),(y_1,z_2)] \Leftrightarrow y_1 \leq z_1, z_1 \leq y_2, y_2 \leq z_2$
(6)	$\mathbf{in}\,[t_1,t_2]\mathbf{thr}\,[(x_1,x_2),(y_1,y_2)] \sqcup \mathbf{in}\,[t_1,t_2]\mathbf{thr}\,[(z_1,z_2),(y_1,y_2)] = \mathbf{in}\,[t_1,t_2]\mathbf{thr}\,[(x_1,z_2),(y_1,y_2)] \Leftrightarrow x_1 \leq z_1, z_1 \leq x_2, x_2 \leq z_2$

Table 2. Axioms for least upper bound of annotations

- The annotated formula $A\,\mathtt{thr}\,R\,\mathtt{in}\,J$ means that there exist(s) *some* time point(s) in the time period J in which A holds throughout the region R. The definition of such a formula in terms of $\mathtt{atp}$ and $\mathtt{at}$ is:

$$A\,\mathtt{thr}\,R\,\mathtt{in}\,J \;\Leftrightarrow\; \exists t \in J.\ \forall s \in R.\ A\,\mathtt{atp}\,s\,\mathtt{at}\,t.$$

  In this case swapping the annotations swaps the universal and existential quantifiers and hence results into a different annotated formula $A\,\mathtt{in}\,J\,\mathtt{thr}\,R$, meaning that for every spatial point in the region R, A holds at some time point(s) in J. Thus we can state *snow* $\mathtt{thr}\,R\,\mathtt{in}\,[jan, mar]$ in order to express that there exists a time period between January and March in which the whole region R is completely covered by the snow. On the other hand *snow* $\mathtt{in}\,[jan, mar]\,\mathtt{thr}\,R$ expresses that from January to March each point of the region R will be covered by the snow, but different points can be covered in different time instants.

2.4 Least Upper Bound and its Constraint Theory

For technical reasons related to the properties of annotations (see [10, 20]), we restrict the rule ($\sqcup$) to least upper bounds that produce valid, new annotations, i.e., rectangular regions and temporal components which are time periods. Thus we consider the least upper bound in the cases illustrated in Table 2.

Axioms (1), (1′), (2) and (2′) allow one to enlarge the region in which a property holds in a certain interval. If a property A holds both throughout a region R_1 and throughout a region R_2 in every point of the time period I then it holds throughout the region which is the union of R_1 and R_2, throughout I.

Notice that the constraints on the spatial variables ensure that the resulting region is still a rectangle. Axioms (3) and (3′) concern the temporal dimension: if a property A holds throughout a region R and in every point of the time periods I_1 and I_2 then A holds throughout the region R in the time period which is the union of I_1 and I_2, provided that I_1 and I_2 are overlapping. By using axiom (4) we can prove that if a property A holds in some point(s) of region R throughout the time periods I_1 and I_2 then A holds in some point(s) of region R throughout the union of I_1 and I_2, provided that such intervals are overlapping. Finally, the last two axioms allow to enlarge the region R in which a property holds in the presence of an `in` temporal annotation.

2.5 Clauses

The clausal fragment of STACLP, which can be used as an efficient spatio-temporal programming language, 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, from 11am to 12am he has breakfast and then in the afternoon 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 (2,12) th [1am,10am].
does(john,eat) atp (2,6) th [11am,12am].
does(john,ski) inr [(500,2000),(1000,2000)] th [12am,4pm].
does(monica,ski) inr [(500,2000),(1000,1500)] th [12am,4pm].
```

The temporal information is represented by a `th` annotation because the property holds throughout the time period. Instead the spatial location is expressed by using an `atp` annotation when the exact position is known, or 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.

```
place(refuge) thr [(700,710),(1200,1205)].
place(ski) thr [(50,2000),(1000,2000)].
```

Below we show how some queries involving the spatial and/or temporal knowledge can be formulated in our language.

1. Where is John between 12am and 2pm?
   `does(john,_) inr R in [12am,2pm]`
   The answer to this query consists of (possibly different) regions where John stays during that time period. We use the `in` annotation because we want to know all the different positions of John between 12am and 2pm while the `inr` annotation allows one to know the region John is in during that time period, even if his exact position is unknown.
   If we asked for `does(john,_) atp R th [12am,2pm]` then we would have constrained John to stay in only one place for the whole time period.
   The query `does(john,_) atp R in [12am,2pm]` asks for definite positions of John sometime in $[12am, 2pm]$.
2. Where is John while Monica is in the refuge?
   `does(john,_) inr R th I, does(monica,_) inr R1 th I, place(refuge) thr R1`
   This query is a composition of a spatial join and a temporal join.

3 Semantics of STACLP

In the definition of the semantics, without loss of generality, we assume all atoms to be annotated with `th`, `in`, `thr` or `inr` labels. In fact, `at` t and `atp` (x, y) annotations can be replaced with `th` $[t, t]$ and `thr` $[(x, x), (y, y)]$ respectively by exploiting the (`at th`) and (`atp thr`) axioms. Moreover, each atom in the object level program which is not two-annotated, i.e., which is labelled by at most one kind of annotation, is intended to be true throughout the whole lacking dimension(s). For instance an atom A `thr` R is transformed into the two-annotated atom A `thr` R `th` $[0, \infty]$. Constraints remain unchanged.

The meta-interpreter for STACLP is defined by the following clauses:

$$demo(empty). \tag{1}$$

$$demo((B_1, B_2)) \leftarrow demo(B_1), demo(B_2) \tag{2}$$

$$demo(A\ \alpha\beta) \leftarrow \alpha \sqsubseteq \delta, \beta \sqsubseteq \gamma, clause(A\ \delta\gamma, B), demo(B) \tag{3}$$

$$\begin{aligned} demo(A\ \alpha'\beta') \leftarrow\ & \alpha_1\beta_1 \sqcup \alpha_2\beta_2 = \alpha\beta, \alpha' \sqsubseteq \alpha, \beta' \sqsubseteq \beta, \\ & clause(A\ \alpha_1\beta_1, B), demo(B), demo(A\ \alpha_2\beta_2) \end{aligned} \tag{4}$$

$$demo(C) \leftarrow constraint(C), C \tag{5}$$

A clause $A\,\alpha\beta \leftarrow B$ is represented at the meta-level by

$$clause(A\,\alpha\beta, B) \leftarrow valid(\alpha), valid(\beta) \tag{6}$$

where *valid* is a predicate that checks whether the interval or the region in the annotation is not empty.

The first two clauses are the ordinary ones to solve the empty goal and a conjunction of goals. The resolution rule (clause (3)) implements both the Modus Ponens rule and the rule ($\sqsubseteq$). It states that given a clause $A\,\delta\gamma \leftarrow B$ whose body B is solvable, we can derive the atom A annotated with any annotation $\alpha\beta$ such that $\alpha \sqsubseteq \delta$ and $\beta \sqsubseteq \gamma$. Such constraints are processed by the constraint solver using the constraint theory for temporal and spatial annotations shown in Sect. 2.2. Clause (4) implements the rule ($\sqcup$) (combined with Modus Ponens and rule ($\sqsubseteq$)). It states that if we can find a clause $A\alpha_1\beta_1 \leftarrow B$ such that the body B is solvable, and if the atom A can be proved with annotation $\alpha_2\beta_2$, then we can derive the atom A labelled with any annotation less or equal than the least upper bound of $\alpha_1\beta_1$ and $\alpha_2\beta_2$. The constraint $\alpha_1\beta_1 \sqcup \alpha_2\beta_2 = \alpha\beta$ is solved by means of the axioms defining the least upper bound introduced in Sect. 2.4. Clause (5) manages constraints by passing them directly to the constraint solver.

4 Modelling and 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. Therefore, in order to properly approach several spatio-temporal analysis problems, it is essential to model and represent such objects and their movement in a suitable way.

In this section, we will briefly summarise some of the most common spatio-temporal data models presented in literature, and thus we will introduce the model adopted in this work to represent the movement of spatio-temporal objects, together with its formalisation within the STACLP language.

Spatio-Temporal Data Models

Although a quite large number of ideas were proposed in literature for modelling spatio-temporal data (see for example [3]), in recent years only a few proposals have been widely discussed. Moreover, no general purpose database system does exist which integrates spatial, temporal and thematic information in a satisfactory way.

In what follows, we provide a brief overview of four of the main data models recently discussed in literature.

Parametric 2-spaghetti: Chomicki and Revesz [6] have introduced a new model, called Parametric 2-spaghetti model, that generalises the 2-spaghetti model (able to represent points, lines and polygons) by allowing an interaction between spatial and temporal attributes. Vertex coordinates can now be *linear functions of time*, thus permitting to model continuous movements along time. As an example, a dynamic region r_1, modelled as a dynamic triangle, could be represented by a set of generalised tuples of the following kind:

ID	x	y	x'	y'	x''	y''	$From$	To
r_1	3	3	10	10	20	$11-t$	8	10
r_1	$13-t$	3	10	t	$2t$	$t-9$	10	∞

Notice that the values in the relation are parametric in the time t, whose value ranges, in each tuple, between *From* and *To*. Query languages for the Parametric 2-spaghetti data model have not been defined yet. This model has been used in [5] to animate spatio-temporal objects.

Type lifting: In this approach, proposed by Erwig, Güting, Schneider and Vazirgiannis [7, 14], collections of ***abstract data types*** for spatial values changing over time are defined. Essentially, the authors introduce data types for moving points and moving regions together with a set of operations on such entities. The design of the model for spatio-temporal data is based on a type constructor τ which transforms any given atomic data type α into a type $\tau(\alpha) = \mathit{time} \to \alpha$ where $\mathit{time} = \mathbb{R}$, that is a continuous model of time is supported. In particular this constructor is applied to two spatial data types, *point* and *region*, allowing one to represent two fundamental abstractions *moving point* (*mpoint*) and *moving region* (*mregion*). Then a set of operations are defined on these types, such as the distance $\mathit{mdistance} : \mathit{mpoint} \times \mathit{mpoint} \to \tau(\mathit{real})$, which returns the distance between two moving points at all times. The presented data types can be embedded into any DBMS data model as attributes data types, and the operations be used in queries.

ST-simplexes: In this model, introduced by Worboys [24], each elemental spatial object – called *simplex* and composed by either a single point, finite straight line segment or triangular area – is assigned with a bitemporal element, which is a finite set of pairs of intervals, representing respectively the database and the event times of the object. A complex spatial object, temporally referenced, is called *ST-complex* and it is modelled by a finite set of ST-simplexes, subject to some constraints that we do not report.
The query language provides operators to make the union, difference and intersection of two ST-complexes, and the selection on the temporal and spatial component. Moreover, it is also given a topological operator to return the boundary of an ST-complex.
In the Worboys' model, the temporal and the spatial dimensions are independent. This prevents from describing spatial relations which may be parametric w.r.t time, i.e. w.r.t their evolution, for instance the relationship that exists between time and the area covered by an incoming tide.

Constraints: The idea behind Constraints Databases is to finitely represent infinite relations by means of (finite) sets of constraints over a given domain (the ***linear constraints over rationals*** or ***real polynomial constraints*** is often used as logical theory of constraints). Among the approaches presented in literature dealing with spatial information, Constraints Databases is the only one to provide a uniform data model for both time and space. In fact, if we consider as class of constraints linear polynomials, we can model both temporal and spatial data with efficient implementations.
In particular, in order to allow an efficient evaluation, Grumbach et al. [13] require that the d-dimensional databases to be considered consist

only of relations with a bounded ***orthographic dimension*** $l \leq d$, informally defined as the maximal number of dependent components in the relation. A typical setting consists of relations over $\mathbb{Q}^2 \times \mathbb{Q}$, the first component for space and the second one for time, with no interaction among the two, obtaining an orthographic dimension equal to 2. An example is:

$$object(S,x,y,t) :- y \leq 6, 1 \leq y, 5x - y - 34 = 0, t \leq 8.$$
$$object(S,x,y,t) :- 7 \leq x, x \leq 11, x - 4y - 3 = 0, t > 8.$$

Notice the absence of interaction between time and the two spatial coordinates.

A Trajectory Data Model

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, as in the simple examples depicted in Fig. 1, where the empty and filled circles represent the observations of two different objects. A very basic operation

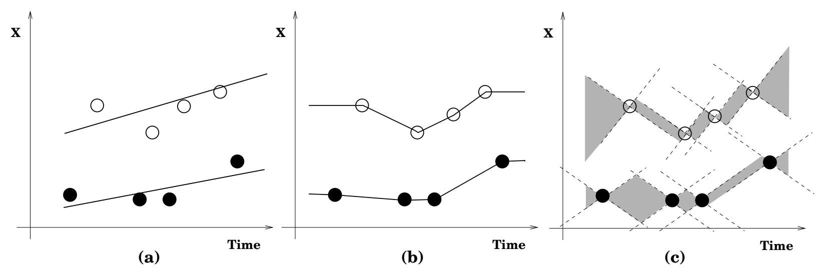

Fig. 1. Raw control points and possible reconstruction of trajectories

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 can distinguish three simple yet interesting categories:

- *Global regression*: if the trajectories are known to follow a simple regular behaviour which can be represented by some analytical time-dependent expression, the movement of an object o can be described by a single regression function. Fig. 1(a) shows an example where a linear regression function is used. Notice that the resulting curve does not always touch all the given control points.
- *Local interpolation*: 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, essentially adhering to Chomicki and Revesz's parametric 2-spaghetti model. Fig. 1(b) depicts a simple example.
- *Domain knowledge-based reconstruction*: in principle, when no interpolation function (global or local) is known, the position of objects out of the control points can be any point in $\mathbb{R}^d$. However, sometimes a "domain knowledge" of a different kind can help to understand how objects behave between observed points, exploiting information about either the objects or the space they move in. Typically, such knowledge is expressed as constraints on the position of objects, which might strongly narrow the range of choices of their movements. A simple example, graphically shown in Fig. 1(c), is given by information on the maximum speed that objects can reach: in this case all positions of an object in space-time must fall within (hyper-)cones having the edge in a control point, axes parallel to the time axes, and aperture representing the maximum speed of the object. Thus, the set of all possible positions of such object between two control points can be obtained as the intersection of the two corresponding (hyper-)cones – corresponding to the gray areas of the picture.

In this work, we will focus on the second proposal, following the parametric 2-spaghetti approach, modelling the movement of objects through linear interpolation between control points and assuming stationarity beyond the extreme control points. As mentioned, this solution yields a good tradeoff between flexibility and simplicity.

Trajectory Representation

In this section we show how trajectories can be modelled in STACLP. Representing trajectories in STACLP has several benefits. 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 an inductive and deductive steps to perform complex kinds of analysis on the behaviour of moving objects as we will show in Sect. 6.

Given an object o the ***observations*** which describe the trajectory of the object is a finite set of triplets (x, y, t), where x and y are the coordinates of the object tracked at time t. As mentioned above, the method adopted in this work to reconstruct the full trajectory from the observations computes the linear interpolation between consecutive localisation points. This kind of interpolation is particularly easy to represent since given two localisation points (x_1, y_1, t_1) and (x_2, y_2, t_2) with no other localisation in the time interval $[t_1, t_2]$ (i.e., the two observations are consecutive), we have that the estimated coordinate at time t, with $t_1 \le t \le t_2$, is (x, y) where:

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

This approximation of trajectories can be easily modelled in STACLP: the localisation points are represented by using the `atp/at` annotation and then the straight line between the two end points is expressed as a constraint.

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 (x2, y2) at t2
⋮
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.

5 An Application Example

Behavioural ecology is the science which studies animal behaviour with special interest in the relation to the environment where animal lives. This can be an interesting application domain for our framework since the problems coped with require the analysis of large spatio-temporal datasets and would definitely benefit from the availability of high level reasoning capabilities.

A typical technique used by biologists to collect information on the studied animals consists of tracking their movements by means of special collars which the animals are equipped with. In this way, large datasets containing spatio-temporal localisations, called *fixes*, of tracked animals are built. 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 to view animals as spatio-temporal objects. The study of their trajectories can be quite helpful in determining the *home range* of animals, i.e. their life area, varying along the time, and it can be useful to establish relationships with other animals in order to understand the modalities of social and spatial aggregation.

The geographical features of the area of interest, such as vegetation, rivers, buildings, roads, which play a basic role in the analysis, are usually stored in a GIS, which thus represents the natural tool support for the biologists. Several commercial GISs offer a support for spatial analyses. For instance ArcView 3.2 provides a number of extensions, such as *Spatial Analyst* [8] by ESRI and *Movement* [15] by USGS Alaska, that implement a set of biological/statistical functions for home range calculations and other spatial analyses applied in animal ecology. However, the lack of tools to analyse in a correlated way space and time makes such systems not completely satisfactory when dealing with these applications.

To exemplify the usefulness of spatio-temporal analysis, we next list a number of relevant problems in behavioural ecology specifically involving spatio-temporal aspects. Such questions emerge from a research leaded by biologists from the University of Siena about the behavioural ecology of crested porcupines in the Maremma Regional Park (Tuscany, Italy) [4]. It is worth recalling that the crested porcupine is mainly nocturnal, lives in natural or artificial burrows and there is very few information available on the behaviour of such species. For this reason there is currently much interest, in the animal ecology field, in studying its habits.

Den Localisation

Crested porcupines are nocturnal: they spend the day inside dens, typically burrows on the ground, whereas during the night they leave their dens looking for food. They usually change dens over time, and it is particularly relevant for researchers to determine when this happens and where the new den is located in order to formulate some hypotheses on the reasons of the displacement as well as to discover unknown behavioural patterns. A technique used by biologists to detect den locations is called *homing-in*. Essentially the researcher physically follows the signal of the radio collar of an animal by means of a portable radio receiver and tries to reach the animal when it is inside the den.

Unfortunately, homing-in is a quite expensive procedure and thus it cannot be applied extensively. Usually it is done on a bi-monthly basis thus creating a two months gap where the location of dens is unknown. Therefore a strong concern consists of understanding where dens are located in the period of time in which there is no homing-in localisation. This can be done exploiting the animals fixes and, in particular, analysing those fixes that have been taken between dawn and sunset, when, according to the expert knowledge the crested porcupine stays inside or close to the den. Given the set of dawn-sunset fixes a probable den can be inferred as a spatial location that has in its neighbourhood a high number of such fixes.

Observe that this kind of analysis requires a correlation of spatial and temporal information since relevant fixes have a specific temporal property (they have been taken between dawn to sunset) and, by using such data, a location which represents a possible den, is determined considering a spatial neighbourhood property such as a high density of fixes.

Relations Among Animals

For understanding the habits and the social behaviour of animals it is extremely relevant to discover the relationships existing among individuals. Notice that this research 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).

To assess the degree of association among individuals, a possibility consists of comparing the overlap of their home ranges, estimated at given time intervals (typically one month or, more generally, k months). However, such method is quite raw because it can happen that animals stay in common areas but in different periods of time or the overlap might be so large that the simultaneous presence of the animals in the area does not ensure that they are really close to each other.

A more precise procedure requires the calculation of the inter-individual distance between animals localised at the same time. We say that 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 localisations are within a certain distance. The effective values for the temporal and spatial thresholds are established by biologists.

Analysing this kind of inter-individual distance between animals, it is possible to make hypotheses about which animals can be considered a couple, form a herd, or avoid other individuals. For instance, two animals of different sex are likely to be a couple in a given period of time if they have a high quantity of contemporary fixes in the considered interval. On the other hand, an animal avoids another one if their inter-individual distance is always greater than a minimal fixed value.

Changes in the Home Ranges

One of the basic objectives in the analysis of animal behaviour is a better understanding of how the animals change their *home ranges* along time. This objective requires, among other tasks, the detection of seasonal variations of the home ranges, both in location and size, and also, to infer the factors determining the dimensions of home ranges for these species.

A simple approach to face this problem consists of partitioning the time period covered by the analysis, $[t_{start}, t_{end}]$, into consecutive time sub-intervals of proper duration ΔT, thus evaluating the home ranges within each sub-interval and analysing the sequence of results obtained. However, in general, determining a suitable value for ΔT results to be not easy. On one hand, it has to be large enough to allow the computation of a significant home range, but, on the other one, it has to be small enough to catch swift changes – *swiftness* being a concept which is highly relative and dependent on the animal species under analysis. In several contexts, such as the study of the crested porcupines, analysts have not enough a priori information on the animals under study to deduce a right time interval. In some cases, moreover, it is even possible that no ΔT value satisfying the two constraints mentioned above exists at all. The definition of ΔT is often subjective and conditioned by our perception of the temporal units (weeks, months, seasons, years), leading to choose temporal intervals which are shifted w.r.t. the observed phenomena. Therefore, there is a clear need for a technique which provides an estimate of home ranges, regardless of the a-priori definition of time intervals.

A simple solution to the above mentioned problem, is to calculate home ranges in a “continuous” way within a given time interval (e.g., a season). The “continuity” can be obtained by replacing the time partitioning approach, described above, with a temporal *sliding window*: now, two parameters δ_t and W_t are defined, and home ranges are computed on overlapping time intervals, each of duration W_t and each shifted of δ_t w.r.t. the previous one, starting from interval $[t_{start}, t_{start} + W_t]$. Notice that the partitioning approach described first, is just a particular case where $W_t = \delta_t = \Delta T$. For example, with $t_{start} =$ “January 1st” and $\Delta T =$ “30 days”, the partitioning approach would result in home ranges computed separately (approximatively) on each month of the year. In the case of animals which change their life areas at a quicker rate (e.g., once a week), the resulting home ranges would mix together several distinct movements, and thus would not provide a sound description of the animal activity. With sliding windows, on the contrary, setting $W_t =$“30 days” and $\delta_t =$ “1 day”, we would obtain the home ranges for the intervals from January 1st to January 31st, from January 2nd to February 1st, and so on. This way, even animal movements at a scale *finer* than the window size produce a detectable change in the home range – the larger and the longer is the animal movement, the larger is the change in the home range, in a continuous way –, and thus distinct movements can be distinguished between each other.

We notice that computing home ranges over a sliding window produces a *smoothly-evolving* home range, which is perfectly analogous to the smooth curve obtained by applying any (of the several well known) sliding window-based smoothing operator on a time series. Such operators have a wide application in several fields, such as noise-reduction filters in signal processing tools. In this sense, the sequence of home ranges obtained for an object o can be viewed as an alternative, improved representation of its trajectory. Therefore, depending on how home ranges are computed and the complexity of their representation, they can be used in some analysis tasks in place of the original trajectory, in order to improve the quality of the results (e.g., in any analysis where the global *trend* of o's movement is relevant, while its single movements are uninteresting and potentially misleading).

Spatio-Temporal Relations Between Localisations and Events

The spatio-temporal relation between an animal and a particular event, defined in time and space, is one of the problems which are often addressed by animal ecology researchers. Indeed very frequently, there is a need for assessing whether and to what extent an event (e.g. a change in a crop cover, a hunting chase, a meteorological or geo-morphological occurrence) defined in time and space, has caused variations in the movement of the monitored animals.

Since the problem requires a co-location analysis task (between animals and events), it presents difficulties which are similar to those mentioned for the localisation of dens. However, since general events have more complex relations with moving animals than simple dens, several additional issues arise.

The first is that spatial and temporal limitations of the effect of the event on animals, must be defined. For instance, we could exclude all events taking place within a spatial distance from individual greater than a fixed threshold or those events occurring later than the occurrence of the localisation itself. Alternatively one could exclude the events occurring before a given period of time.

Therefore, we should expect that if an event concretely influences the animals, it would then "attract" or "reject" them, or in general modify in some way the movement they performed *before* the event, to an extent which is proportional to their spatio-temporal distance.

Moreover this technique should allow us to estimate the time interval between the occurrence of the event and its effects (as well as their end) on the animal.

These general problems allow several different concrete formulations, which can require simple or complex analysis tasks, mainly depending on the assumptions that can be made.

In a simple setting, we can assume that the event has visible effects on the level of single animals (thus not requiring the concept of herd or group), and that such effects can only be of the attraction and rejection types. In this case, the analysis task requires to compute spatio-temporal distances between animals and events, in order to find out which animals might potentially be influenced by which events. Therefore, for each pair *(event, animal)*, the existence of a rejection or attraction movement should be checked, by analysing how the spatial distance between the pair changes after the event.

In a more complex situation, we could be required to view animal groups (e.g., herds) as a whole. This is particularly true when either (i) the behaviour of a single animal is not significant, and thus it is potentially misleading (e.g., it could have an anomalous reaction to the event, not representative of its species or its group); or (ii) we are simply interested in comparing the behaviour of different herds; or (iii) the event influences the animals by altering some properties of the group which are not evident by looking at its single members (e.g., the event could induce the group to compact itself, as a self-defensive reaction, even though its members movement is not sensibly altered). In this case, the notion of animal group has to be defined and computed, prior to evaluate the spatio-temporal nearness between animals (considering each group as a whole) and events. Then, the candidate groups, i.e. those which were close enough to an event to be potentially influenced, are analysed in order to assess which ones show a behavioural alteration w.r.t. their corresponding events. The latter step will require a suitable formalisation of "behaviour change", whose complexity can greatly vary depending on the properties it analyses.

6 Support for Spatio-Temporal Analysis in STACLP

In the previous section we have described some examples of analysis of spatio-temporal data which are relevant in a specific application domain. Generally speaking a framework for dealing with spatio-temporal (but also general data) should offer the ability of querying such data and of reasoning over them.

Current systems allow the user to exploit some consolidated mechanisms such as SQL-like queries, statistical functions and some spatial operations. Despite the fact that these systems provide efficient and mature technology to query standard relational or spatial data, they do not offer the user high level operations to handle the special time and space related dimensions. Indeed, nowadays the development of a spatio-temporal application with commercial systems requires the programming of some ad-hoc components by means of procedural languages provided by the system itself for customisation purposes. However, the absence of specific spatio-temporal operations makes the customisation of applications extremely complex, especially for non-expert users.

In our opinion, more sophisticated mechanisms are needed to provide the user with high-level and, at the same time, more general support for reasoning. We believe that the language for "programming" the extensions should be a real knowledge representation language, or, better to say, a very high level query language as much as the last extensions to SQL are a declarative programming language for extending relational databases. This view is shared by a number of other researchers, who all work on the integration of declarative paradigms and GISs [13, 25, 23].

Our proposal is mainly based on the use of computational logics as knowledge representation formalisms and as inference engine supporting reasoning.

In particular, in our framework both *deductive* and *inductive* inferences are conciliated as well as enriched with spatio-temporal primitive operations. 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. On the other hand, induction is in some sense the inverse of deduction, that is deriving general principles from concrete data, therefore synthesising new knowledge from data or experience.

Next subsection focuses on deduction, whereas Sect. 7 is devoted to introduce mechanisms to support inductive analysis and it describes how these tools can be successfully used in our case study.

6.1 Deductive Analysis in STACLP

In this section we will show how the deductive capability of our language allows us to solve several problems presented in Sect. 5, namely, finding out the estimated position of the den whenever its real location is unknown, discovering the animals which are likely to be a couple and determining how the home range of a given animal changes.

As described in Sect. 4, we model the spatio-temporal localisations of each crested porcupine by 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 some of the questions proposed by the biologists. The rules are slightly simplified by removing some implementation details, like the use of the predicate *constraint* which indicates that the atom is a constraint. Such a presentation allows us 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`.

Let us now formalise the questions we want to cope with. From the expert knowledge we know that the crested porcupine stays in its den during the day whereas it usually spends the night far from it. Hence, in order to determine the position of the den we collect all the fixes of the animal ranging between an hour before dawn and an hour after sunset, the idea being that in this time period the animal is probably close to its den. The rules that implement our analysis criteria are the following:

```
possible_loc(Id,Lloc) at T :-
   findall(loc(X,Y), (fix(Id) atp(X,Y) at T1, dawn_sunset(T1) at T),
         Lloc).

dawn_sunset(T1) at T :- light(D,S) at T, D-3600 <= T1, T1 <= S+3600.

prob_den(Id,Rad,Prob,L) in [T1,T2] :- possible_loc(Id,Lloc) in [T1,T2],
                                       neighbour_list(Lloc,Rad,Prob,L).
```

The first clause returns the list of positions `Lloc` of the animal `Id` between dawn and sunset in a certain day. The predicate `dawn_sunset`, occurring in this rule, checks whether the time `T1` of the localisation falls between one hour before dawn and one hour after sunset of the day `T` (i.e., the day time T belongs to). Notice that, since the dawn and sunset times vary along the year, the predicate `light` is used to record a monthly estimate of such times.

The third clause extracts from the list of the localisations computed by the predicate `possible_loc` the positions which are likely to be dens. The predicate `neighbour_list` (not defined for sake of brevity) selects those positions whose neighbourhood (with radius `Rad`) includes a great quantity of fixes between dawn and sunset (precisely greater than the value `Prob`). This quantity is estimated by considering the ratio between the number of fixes in the neighbourhood and the total number of fixes for the given animal in the period of time of interest.

Consider now the second problem illustrated in Sect. 5, namely providing a high level mechanism to determine relations among individuals. 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 likely couples we exploit the notion of contemporary fixes: two fixes are contemporary if they are within a certain distance and in a certain time interval. We recall that the effective values for the threshold have to be chosen by the domain experts.

At this point we compute the number of contemporary fixes for a pair of animals and according to this number 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. 5.

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 [18, 21].

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`.

It is worth noticing how the use of a declarative language in general, and STACLP in particular, which supports spatio-temporal reasoning, allows for a simple, compact and readable formalisation of the expert knowledge as well as it provides a very expressive query language. Moreover, STACLP offers the ability of performing external calls to ad-hoc functions provided by independent applications. These features make the STACLP code a kind of abstraction level over the "low level" tools used by domain experts. Actually, we do not intend STACLP as a framework to replace existing specialised systems, on the contrary, it should be integrated with these systems, thus it can be considered as a "middleware" in which very sophisticated spatio-temporal reasoning can be performed.

7 Inductive Analysis in STACLP

As seen in the previous section, deductive reasoning can be useful to solve several analysis problems which essentially require to find entities and values having some, possibly complex, properties. The STACLP language already showed to have the expressive power needed to fully formalise – and thus to solve – such problems. Moreover, thanks to its high-level nature, in most cases the formalisation step is simple and intuitive, yielding a very user-readable STACLP program.

However, dealing with more complex analysis tasks, it is quite common to meet concepts and abstract entities whose definition through deductive rules can be extremely difficult. It is especially the case when only vague, ambiguous or incomplete formalisations of the concept are available. A simple example, already introduced in this chapter, is the notion of animal group, or herd. While its meaning is intuitive and clear from a human view-point, formalising it can be difficult and highly dependent on the specific application – that is to say, a general definition of herd, suitable for any context, does not exist. In many cases, a suitable solution to this problem requires the *extrapolation* of new 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 a solution for some of the more complex problems introduced in Sect. 5.

7.1 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. The greater the similarity within the group, and the greater the difference between groups, the better or more distinct the clustering. In our context, the aim is to partition the moving objects which populate our database, especially animals, into groups of individuals following similar trajectories.

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, described by the following pseudo-code, is essentially an iterative convergence process which tries to find "stable" centres:

K-means

```
1. Select k random points as initial centres;
2. REPEAT
3.   for all points $p$
4.      Assign $p$ to its closest centre;
5.   for all clusters $C$
6.      Re-compute the centre of $C$;
7. UNTIL no centre changed in last iteration or max n. iterations reached;
```

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 most straightforward and most common way of computing cluster centres in standard contexts where objects are described by vectors of basic attributes (e.g., the coordinates of points in a 2-dimensional space) is to define the attributes of the centre as the average of the values taken by the objects in the cluster. The extension of this idea to the spatio-temporal context is quite simple, since trajectories are essentially moving points, thus providing a very natural and intuitively sound instantiation of the centre computation: a centre will be an object whose position at time t is the average of the positions of all objects in the cluster at time t.

On the contrary, it is more difficult to find a single most natural instantiation of the distance operation between objects, although some definitions exist which can cover some common concrete problems. For this reason, we first provide the STACLP rules which define a k-means algorithm for trajectories, independently of the distance definition. Later on, a set of possible choices for the distance function is given, together with the implementation of one of them.

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(Iter, [], []).
assign(0, [A1|A], [Obj1|Objs]) :- K=random(k),
          A1=cluster(Obj1,K), assign(0, A, Objs).
assign(Iter, [A1|A], [Obj1|Objs]) :- Iter>0,
             closest(Iter-1, Obj1, Cluster),
             A1=cluster(Obj1,Cluster), assign(Iter, A, Objs).
```

The objs_to_cluster predicate defines the set of objects to be clustered, which can be easily customised by rewriting the body of its rule with any suitable criterion (here we selected all objs(X) objects having some fixes defined). For any iteration Iter, 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 Sect. 6.1). 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 [19] (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 [19]. 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.
```

7.2 Knowledge Discovery on Trajectories

Some of the problems listed in Sect. 5 are difficult to solve with the only mean of deduction tools, while others could benefit from other capabilities, such as induction, to yield a more satisfactory solution. In this section we provide two examples where using the induction tool introduced above helps to find a simple and compact solution to two of the above mentioned problems: detecting the relations between animals and the relations between localisations and events.

In the first case, a fully deductive approach has already been presented in Sect. 6.1, 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 noticing that animal couples and animal herds are 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(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 an 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).

The problem of studying the relations between localisations and events requires to discover the effects produced on the behaviour of animals by an event. A natural way to face the problem is to compare the behaviour of the animals *before* the event with their behaviour *after* the event. As mentioned in Sect. 5, several approaches can be followed, analysing either the behaviour of single individuals, or the behaviour of herds. The individual-based class of solutions simply requires to specify a suitable notion of "behaviour", in order to be able to quantify behaviour changes. In what follows, we provide two examples of solutions falling in the (more complex) herd-based class, which exploits both the deductive engine of STACLP and the clustering tool defined at the beginning of this section.

A first formalisation of the event-effects problem can be given in terms of herd dispersion/compactness, saying that a herd reacts to the event if and only if its dispersion, measured as the sum of distances between individuals (or, equivalently, as the average of distances), grows beyond a given threshold factor `max_disperse` or shrinks beyond a factor `max_compact`. This can be implemented as a STACLP program performing a clustering of the animals before the event and checking growth and shrink factor after the event.

As a first step, for each object `objs(X)` two new objects are created: `before(X)` and `after(X)`, obtained by cutting the trajectory of `obj(X)` respectively on the time intervals $[t_{start}, t_{event}]$ and $[t_{event}, t_{end}]$, where t_{event} is the instant when the event begins. This operation can be obtained by means of STACLP rules similar to those provided in the previous example for the `cut_obj(X)` objects.

Next, a clustering on the `before(X)` objects has to be performed, which can be achieved, as in the previous example, by simply rewriting the definition of the `objs_to_cluster` predicate in the following way:

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

Finally, for each cluster the cluster dispersion – i.e., the sum of distances between the objects in the cluster – is computed before and after the event, tagging each cluster as "dispersed" or "compacted" if the dispersion respectively grows or shrinks beyond the thresholds `max_disperse` and `max_compact`.

```
cluster_size(K, Before, After):- cluster(K,O_list),
                findall( (D_before, D_after),
                         ( member(before(X1), O_list),
                           member(before(X2), O_list),
                           X1 != X2,
                           distance(before(X1), before(X2), D_before),
                           distance(after(X1), after(X2), D_after),
                         D_list),
                sum_pairs(D_list, (Before, After)).
dispersed_cluster(K) :- cluster_size(K, Before, After),
                        After > Before * max_disperse.
compacted_cluster(K) :- cluster_size(K, Before, After),
                        Before > After * max_compact.
```

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).

8 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). Moreover we showed 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. When dealing with either spatial objects, or temporal objects, or spatio-temporal objects in order to apply the above data mining techniques, the most crucial and difficult problem is to find a clear definition of the basic concept of *item*, which is the elementary unit of data to analyse, and the derived concept of *transaction*, which is a collection of items. So far, it seems to us that the problem is quite depending on the kind of application at hand, but one of our future research goals is to establish definitions abstract enough for these concepts in order to capture most of the possible applications.

Another promising direction we addressed concerns qualitative spatial reasoning. Some preliminary results are presented in [22] and we would like to analyse forms of qualitative reasoning on trajectories.

Acknowledgements: We thank T. Ceccarelli and A. Massolo for providing us the case study and F. Fornasari and B. Furletti who collaborated in the implementation of the system. This work has been partially supported by European Project IST-1999-14189 - Rev!gis.

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. Abraham and J.F. Roddick. Survey of Spatio-Temporal Databases. *GeoInformatica*, 3(1):61–99, 1999.

4. 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.
5. J. Chomicki, Y. Liu, and P.Z. Revesz. Animating Spatiotemporal Constraint Databases. In *Spatio-Temporal Database Management*, volume 1678 of *LNCS*, pages 224–241. Springer, 1999.
6. J. Chomicki and P.Z. Revesz. Constraint-Based Interoperability of Spatiotemporal Databases. *GeoInformatica*, 3(3):211–243, 1999.
7. M. Erwig, R. H. Güting, M. Schneider, and M. Vazirgiannis. Spatio-Temporal Data Types: An Approach to Modeling and Querying Moving Objects in Databases. *GeoInformatica*, 3(3):269–296, 1999.
8. ESRI. Spatial Analyst. http://www.esri.com.
9. 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.
10. T. Frühwirth. Temporal Annotated Constraint Logic Programming. *Journal of Symbolic Computation*, 22:555–583, 1996.
11. S. Gaffney and P. Smyth. Trajectory clustering with mixture of regression models. In *KDD Conf.*, pages 63–72. ACM, 1999.
12. S. Grumbach, P. Rigaux, and L. Segoufin. The DEDALE System for Complex Spatial Queries. In *SIGMOD Conf.*, pages 213–224. ACM, 1998.
13. S. Grumbach, P. Rigaux, and L. Segoufin. Spatio-Temporal Data Handling with Constraints. *GeoInformatica*, 5(1):95–115, 2001.
14. R.M. Güting, M.H. Böhlen, M. Erwig, C.S. Jensen, N. Lorentzos, M. Schneider, and M. Vazirgiannis. A Foundation for Representing and Querying Moving Objects. *TODS*, 25(1):1–42, 2000.
15. P. N. Hooge, W. M. Eichenlaub, and E. K. Solomon. Animal Movement Extension to ArcView. Ver. 2.0., 1997. Alaska Biological Science Center, U.S. Geological Survey, Anchorage, AK, USA.
16. P.C. Kanellakis, G.M. Kuper, and P.Z. Revesz. Constraint query languages. *Journal of Computer and System Sciences*, 51(1):26–52, 1995.
17. A. Ketterlin. Clustering sequences of complex objects. In *KDD Conf.*, pages 215–218. ACM, 1997.
18. P. Mancarella, A. Raffaetà, C. Renso, and F. Turini. Integrating Knowledge Representation and Reasoning in Geographical Information Systems. *International Journal of GIS*. To appear.
19. M. Nanni. *Clustering Methods for Spatio-Temporal Data*. PhD thesis, Dipartimento di Informatica, Università di Pisa, 2002.
20. A. Raffaetà and T. Frühwirth. Spatio-Temporal Annotated Constraint Logic Programming. In *PADL'01*, volume 1990 of *LNCS*, pages 259–273. Springer, 2001.
21. A. Raffaetà, C. Renso, and F. Turini. Enhancing GISs for Spatio-Temporal Reasoning. In *GIS'02*, pages 35–41. ACM, 2002.
22. 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.
23. S. Spaccapietra, editor. *Spatio-Temporal Data Models & Languages (DEXA Workshop)*. IEEE Computer Society Press, 1999.
24. M. F. Worboys. A unified model for spatial and temporal information. *The Computer Journal*, 37(1):26–34, 1994.
25. M. F. Worboys. *GIS - A Computing Perspective*. Taylor & Francis, 1995.