Research in Probabilistic Spatiotemporal Databases: The SPOT Framework

John Grant, Francesco Parisi, and V.S. Subrahmanian

Abstract. We start by providing an overview of research on probabilistic spatiotemporal databases. The bulk of the paper is a review of our previous results about probabilistic spatiotemporal databases using the SPOT approach. Presently these results are scattered in various papers and it is useful to provide a uniform overview. We also present numerous interesting research problems using the SPOT framework for probabilistic spatiotemporal databases that await further work.

1 Introduction

In recent years much interest has focused on the tracking of moving objects and reasoning about moving objects. Particularly with GPS systems it becomes possible to track vehicles, cell phones, supply items, RFID tags, and the importance of such tracking continues to increase. Clearly, space and time are important factors in this endeavor but probability is also useful. The reason for that is that the locations, and possibly the identity of the objects and the time may be known only with some uncertainty and we can express such uncertainty by the use of probability. Furthermore, we claim that in many cases the probability itself is not known exactly, in which case the use of a probability interval is appropriate. Thus our framework involves space, time, and probability intervals.

John Grant
Towson University, Towson, Maryland, USA, and
University of Maryland, College Park, Maryland, USA
e-mail: jgrant@towson.edu

Francesco Parisi
Università della Calabria, Rende (CS), Italy
e-mail: fparisi@deis.unical.it

V.S. Subrahmanian
University of Maryland, College Park, Maryland, USA
e-mail: vs@cs.umd.edu

Z. Ma & L. Yan (Eds.): *Advances in Probabilistic Databases*, STUDFUZZ 304, pp. 1–22.
DOI: 10.1007/978-3-642-37509-5_1

A couple of years ago we became interested in providing a formalism for the representation, querying, and updating of probabilistic spatiotemporal databases in a straightforward manner with a simple syntax and an intuitive model-theoretic semantics. For this purpose we introduced the SPOT (Spatial PrObabilistic Temporal) framework in which a basic statement (atomic formula) states that a particular object is in a particular region (for location) at a particular time with a probability that is in a particular probability interval.

We then implemented the SPOT formalism and tackled several issues concerning SPOT databases. In particular, we showed that several problems involving SPOT databases can be transformed into linear programming problems. We developed indexing methods to speed several types of selection queries. We also studied updating SPOT databases in the spirit of AGM-style belief revision. Altogether we published five papers on the SPOT approach. The purpose of this paper is to summarize our results in one place and to propose numerous research questions for the further study of probabilistic spatiotemporal databases in the SPOT framework.

We start in Section 2 by reviewing some of the important work done by other researchers related to probabilistic spatiotemporal databases that did not use the SPOT approach. Then Section 3 introduces the syntax and semantics of SPOT databases and provides an example that will be used throughout the paper to illustrate various concepts. Section 4 contains the fundamental results about SPOT databases including the transformation to linear programming. The processing of certain kinds of selection queries is discussed in Section 5; both optimistic and cautious answers are considered. Section 6 reviews the algorithms for database updating/revision. Finally, Section 7 contains interesting research questions.

2 Review of Previous Related Research

In order to place the SPOT framework in the proper context we now review research related to it. This is by no means an exhaustive survey; we give a historical background and briefly review some important papers; many additional references appear in the works we mention. The problem of predicting where moving objects will be in the future, when they will be there, and with what probability is intrinsically challenging. This can be easily understood by observing that reasoning with just one or two of these three aspects (probability, space and time) is already quite challenging. In fact, researchers have investigated separately probabilistic databases, probabilistic spatial databases, probabilistic temporal databases, and probabilistic spatiotemporal databases.

Kiessling and his group [28] develop the DUCK framework for reasoning with uncertainty. They provide an elegant, logical, axiomatic theory for uncertain reasoning in the presence of rules. In the same spirit as Kiessling et al., Ng and Subrahmanian [45] present a probabilistic semantics for deductive databases — they assume absolute ignorance, and furthermore, assume that rules are present in the system. Lakshmanan and Sadri [38] show how selected probabilistic strategies can be used to extend the previous probabilistic models. Lakshmanan and Shiri [39]

demonstrate how deductive databases may be parameterized through the use of conjunction and disjunction strategies. Barbara et al. [3] develop a probabilistic data model and propose probabilistic operators. Their work is based on the assumption that probabilities of compound events can always be precisely determined. Cavallo and Pittarelli [8, 54], propose a model for probabilistic relational databases in which tuples in a probabilistic relation are interpreted using an exclusive or. Dey and Sarkar [15] propose an elegant 1NF approach to handling probabilistic databases. In another work Kifer and Li [33] examine quantitative logic programming and introduce formal semantics for such systems. Other systems from the probabilistic database community also provide insight into probabilistic information reasoning and storage [18, 19, 13, 34, 31]. Lukasiewicz and his colleagues [41, 42] study probabilistic reasoning in logic programming, as does Dekhtyar [14]. However, none of these works explicitly handle space or time.

The problem of efficiently storing and querying data representing spatial predictions has deserved much attention by many researchers. Tao et al [60] develop an indexing structure for spatial probabilistic data designed to solve a specific problem. They assume that there is a single probability distribution function detailing where an object might be at a given point in time in the entire space and their focus is on optimizing access to that probability density function. Using an R-tree-inspired U-tree indexing structure they use hyperplanes to approximate the evolution of these probabilistically constrained regions between time points. In [46], methods for dealing with positionally uncertain spatial data are considered. Their data model associates each point with a cluster, where points in the same cluster have the same error. This model also allows only one pdf. The authors describe a PrR-tree for storing and querying survey data, which uses a rectangular bounding region whose corners are defined via Gaussian distributions. Like the above work, in [40], Lian et al use a data model with one pdf over each object's locations. They introduce probabilistic group nearest neighbor queries, where given a set of points and a probability threshold the system returns the set of objects that have minimal aggregate distance to the set of points with a probability over the threshold. Dai et al [12] focus on probabilities for the *existence* of a given object at a given point without worrying about the possibility of the object being at another point. They show how to build an augmented R-tree and use that tree to answer selection queries more effectively than considering probability as an extra dimension in the R-tree. [7] uses a paradigm called "line simplification" to approximate trajectories of moving objects, though their framework does not involve uncertainty.

On the purely temporal side, Snodgrass is one of the first to model indeterminate instances [59] — he proposes the use of a model based on three valued logic. Dutta [17] later give a fuzzy logic based approach to handle *generalized temporal events* — events that may occur multiple times. This approach is also used by Dubois and Prade [16]. Gadia [25] proposes an elegant model to handle incomplete temporal information as well. He models values that are completely known, values that are unknown but are known to have occurred, values that are known if they have occurred, and values that are unknown even if they occurred. Koubarakis [36] proposes the use of constraints for representing temporal data. Brusoni [6] develops a

system called LaTeR that restricts constraints to conjunctions of linear inequalities, as does Koubarakis' work. [4] develops a framework to track uncertainty, time, and the pedigree of data, but does not handle spatial information. As a matter of fact none of these works handle both space and time.

There is also much work on *spatio-temporal logics* [24, 43] in the literature. These logics extend temporal logics to handle space. This includes work on qualitative spatio-temporal theories (for a survey see [11, 44] [58] which discusses the frame problem when constructing a logic-based calculus for reasoning about the movement of objects in a real-valued co-ordinate system). [55] focuses on relative position and the orientation of objects with existing methods for qualitative reasoning in a Newtonian framework. Other efforts combine a spatial logic, such as $RCC-8$ [56], $BRCC-8$ [62] and $S4_u$ [5], with propositional temporal logics ($\mathscr{PTL}$). The work on spatio-temporal reasoning is mostly qualitative [11, 43, 63, 23], and focuses on relations between spatio-temporal entities while dealing with discrete time. However, these works are not intended for reasoning about moving objects whose location at a given point in time (past, present or future) is uncertain (they not consider probabilities).

In addition to the above works on spatio-temporal logics, there are works on logics integrating time and probabilities. Much of this work was performed in the model checking community. The PRISM system [37] supports a mix of time and probabilities for model checking with respect to specifications written in the temporal probabilistic logics PCTL [30] and CSL [2]. However, none of these works has any spatial element in them, and they focus on model checking, not on handling knowledge bases. The work on "go" theories [20, 22, 21] focuses on spatio-temporal logical theories that are sets of "go" atoms. Such atoms intuitively describe known plans of moving objects. A go-atom states that an object will go from location A to location B, leaving A at a time point in some time interval, arriving at B at a time point in some interval, and traveling in the interim at a velocity within some stated interval. [20] develops a basic theory of "go" theories, while [22] gives a closed world assumption for such theories. Later, [51] extends this logic to include some probabilistic information about such plans. Finally, the SPOT framework extends this work to uncertainty about where objects might be at a given time [50, 49].

While there is substantial work in indexing spatial temporal data without probabilities [52, 61, 35, 1, 53, 29], none of these works address a data model compatible with our SPOT framework: they suppose no probabilities and model object movement as linear. SPOT databases were developed by the authors in past work [50, 49] to store such predictions without making the assumptions in prior work[60, 12, 7, 4]. In fact, our observations about the advantages of the SPOT approach hold also for some very recent papers we mention in the next paragraph.

Up to this point we reviewed work done prior to our formulation of SPOT databases. We end this section with a brief review of some recent papers on probabilistic spatiotemporal databases. These papers do not use the SPOT approach. Chung et al. [10] derive a pdf for the location of an object moving in a one-dimensional space by using its past moving behavior or the moving velocity distribution. Their probabilistic range queries find objects that are in a specified region

of space within a specified time interval, and with a probability that is at least a threshhold value. Their indexing takes care of eliminating object that are too far from consideration. Zhang et al. [65] provide a framework that allows their model to be incorporated easily into existing DBMSs and work for all objects even if their location and velocity are uncertain and the movements are unusual. For indexing they use a B^x-tree which is a variant of a B^+-tree that is applicable to moving objects. Yang et al. [64] work with moving objects in indoor space which is different from Euclidean and spatial network spaces. Their query asks for all sets of k objects that have at least a threshold probability of containing the k nearest objects to a given object. This paper defines a minimal indoor walking distance and uses two types of pruning to efficiently solve such queries. Chen et al. [9] deal with a similar problem but use a TPR-tree for indexing. They also deal with the query result quality by using both a false positive and a false negative rate. Finally, the most recent paper we mention, [66] deals primarily with objects moving along road networks, certainly an important application. They introduce a novel indexing mechanism called UTH (Uncertain Trajectories Hierarchy) for efficiently processing probabilistic range queries.

3 A SPOT Database

This section reviews the syntax and semantics of SPOT databases given in [50].

Syntax

We assume the existence of a set ID of objects ids, a set T of time points ranging over the integers, and a finite set $Space$ of points. Unless stated otherwise, we assume that $Space$ is a grid of size $N \times N$ where we only consider integer coordinates [1]. We assume that an object can be in only one location at a time, but that a single location may contain more than one object.

A *rectangle* is any region in $Space$ that can be described by constraints of the form $left \leq x \leq right$ and $bottom \leq y \leq top$ where $left, right, bottom, top$ are integers in $[0..N]$. Thus, all rectangles have edges parallel to the x and y-axes. A rectangle is empty if either $left > right$ or $bottom > top$.

Definition 1 (SPOT **atom/database).** A SPOT *atom* is a tuple $(id, r, t, [\ell, u])$, where $id \in ID$ is an object id, $r \subseteq Space$ is a non-empty rectangular region in the space, $t \in T$ is a time point, and $\ell, u \in [0,1]$ are probability bounds with $\ell \leq u$. A SPOT *database* is a finite set of SPOT atoms.

Intuitively, the SPOT atom $(id, r, t, [\ell, u])$ says that objects id is/was inside the specified region r at time t with probability in the $[\ell, u]$ interval.

[1] The framework is easily extensible to higher dimensions.

Example 1. Consider a lab where data coming from biometric sensors are collected, analyzed and stored. Biometric data such as faces, voices, and fingerprints recognized by sensors are matched against given profiles (such as those of people having access to the lab) and tuples like those in Table 1 are obtained. Every tuple consists of the profile id resulting from the matching phase, the area of the lab where the sensor recognizing the profile is operating, the time point at which the profile has been recognized, and the lower and upper probability bounds of the recognizing process getting the tuple. For instance, the tuple in the first row of Table 1 representing the SPOT atom $(id_1, d, 1, [0.9, 1])$ says that profile having id id_1 was in region d at time 1 with probability in the interval $[0.9, 1]$. In Figure 1, the plan of the lab and the areas covered by biometric sensors are shown. In area d a fingerprint sensor is located, whose high accuracy entails a narrow probability interval with upper bound equal to 1. After fingerprint authentication, the id_1 profile was recognized at time 3 in areas b and c with probability in $[0.6, 1]$ and $[0.7, 0, 8]$, respectively.

Table 1 SPOT database $\mathscr{S}_{lab}$

Id	Area	Time	Lower Prob	Upper Prob
id_1	d	1	0.9	1
id_1	b	3	0.6	1
id_1	c	3	0.7	0.8
id_2	b	1	0.5	0.9
id_2	e	2	0.2	0.5
id_2	e	3	0.6	0.9

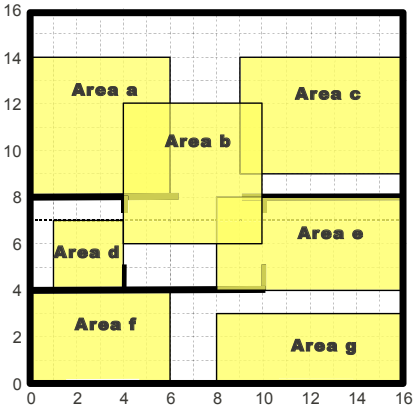

Fig. 1 Areas of the lab

Given a SPOT database $\mathscr{S}$, a fixed object *id* and a fixed time t, we use the notation $\mathscr{S}^{id,t}$ to refer to the set:

$$\mathscr{S}^{id,t} = \{(id', r', t', [\ell', u']) \in \mathscr{S} \mid id' = id \wedge t' = t\}.$$

Semantics

The meaning of a SPOT database is given by the set of interpretations that satisfy it.

Definition 2 (SPOT interpretation). A SPOT *interpretation* is a function $I : ID \times Space \times T \rightarrow [0,1]$ such that for each $id \in ID$ and $t \in T$,

$$\sum_{p \in Space} I(id, p, t) = 1.$$

For a given an interpretation I, we sometimes abuse notation and write $I^{id,t}(p) = I(id, p, t)$. In this case, $I^{id,t}$ is a probability distribution function (PDF).

Example 2. Interpretation I_1 for the SPOT database $\mathscr{S}_{lab}$ introduced in Example 1 is as follows.

$$\begin{array}{ll}
I_1(id_1,(3,6),1) = 0.4 & I_1(id_1,(2,5),1) = 0.2 \\
I_1(id_1,(3,5),1) = 0.3 & I_1(id_1,(5,5),1) = 0.1 \\
I_1(id_1,(7,5),2) = 0.5 & I_1(id_1,(4,2),2) = 0.5 \\
I_1(id_1,(10,10),3) = 0.7 & I_1(id_1,(7,5),3) = 0.3 \\
I_1(id_2,(8,10),1) = 0.1 & I_1(id_2,(12,12),1) = 0.9 \\
I_1(id_2,(9,7),2) = 0.3 & I_1(id_2,(12,13),2) = 0.7 \\
I_1(id_2,(14,5),3) = 0.8 & I_1(id_2,(12,14),3) = 0.2
\end{array}$$

Moreover, $I_1(id, p, t) = 0$ for all triplets (id, p, t) not mentioned above.

Given an interpretation I and region r, the probability that object *id* is in r at time t *according to* I is $\Sigma_{p \in r} I(id, p, t)$. We now define satisfaction by an interpretation.

Definition 3 (Satisfaction). Let $a = (id, r, t, [\ell, u])$ be a SPOT atom and let I be a SPOT interpretation. We say that I satisfies a (denoted $I \models a$) iff $\sum_{p \in r} I(id, p, t) \in [\ell, u]$. I satisfies SPOT database $\mathscr{S}$ (denoted $I \models \mathscr{S}$) iff I satisfies every atom in $\mathscr{S}$.

Example 3. Continuing with our running example, interpretation I_1 satisfies the SPOT atom $(id_1, d, 1, [0.9, 1])$ as, for id id_1 and time point 1, I_1 assigns probability 0.4 to $(3,6)$, 0.3 to $(3,5)$, and 0.2 to $(2,5)$ (which are points in area d), and probability 0.1 to $(5,5)$ which is outside area d. Hence, the probability that id_1 is in area d at time point 1 is 0.9, which belongs to the interval $[0.9, 1]$ specified by the considered SPOT atom. Reasoning analogously, it is easy to see that I_1 satisfies all of the atoms in Table 1 except for $(id_2, b, 1, [0.5, 0.9])$, as the probability to be in area b at time 1 for id id_2 is set to 0.1 by I_1, instead of a value in $[0.5, 0.9]$.

Let I_2 be an interpretation equal to I_1 except that $I_2(id_2, (8,10), 1) = 0.7$ and $I_2(id_2, (12,12), 1) = 0.3$. This interpretation satisfies the SPOT database $\mathscr{S}_{lab}$.

4 The Basic Definitions and Results

We use $\mathbf{I}(\mathscr{S})$ to denote the set of interpretations that satisfy a SPOT database $\mathscr{S}$, that is, $\mathbf{I}(\mathscr{S}) = \{I \mid I \models \mathscr{S}\}$.

Definition 4 (Consistency). A SPOT database $\mathscr{S}$ is *consistent* iff $\mathbf{I}(\mathscr{S}) \neq \emptyset$.

Definition 5 (Compatibility). A SPOT atom a is compatible with a SPOT database $\mathscr{S}$, denoted as $a \Subset \mathscr{S}$, iff $\mathscr{S} \cup \{a\}$ is consistent.

Definition 6 (Entailment). A SPOT database $\mathscr{S}$ *entails* a SPOT atom a, denoted as $\mathscr{S} \models a$, iff $\forall I \in \mathbf{I}(\mathscr{S})$, $I \models a$. A SPOT database $\mathscr{S}_1$ entails a SPOT database $\mathscr{S}_2$, denoted as $\mathscr{S}_1 \models \mathscr{S}_2$, iff $\forall a \in \mathscr{S}_2$, $\mathscr{S}_1 \models a$.

Example 4. Interpretation I_2 of Example 3 proves that the SPOT database $\mathscr{S}_{lab}$ of Example 1 is consistent. It is easy to see that $a = (id_1, f, 2, [0, 0.5]) \Subset \mathscr{S}_{lab}$ because $I_2 \models \mathscr{S}_{lab} \cup \{a\}$ and $\mathscr{S}_{lab} \models (id_1, d, 1, [0.75, 1])$.

Given a SPOT database $\mathscr{S}$, an object $id \in ID$, and a time point $t \in T$, that is, $\mathscr{S}^{id,t}$, [50] defined a set $LC(\mathscr{S}, id, t)$ of linear constraints. $LC(\mathscr{S}, id, t)$ uses variables v_p to denote the probability that object id will be at point $p \in Space$ at time t.

Definition 7 ($LC(\cdot)$). For SPOT database $\mathscr{S}$, $id \in ID$, and $t \in T$, $LC(\mathscr{S}, id, t)$ contains:

- $\forall (id, r, t, [\ell, u]) \in \mathscr{S}^{id,t}$,
  $\left(\sum_{p \in r} v_p \geq \ell\right) \in LC(\mathscr{S}, id, t)$,
  $\left(\sum_{p \in r} v_p \leq u\right) \in LC(\mathscr{S}, id, t)$
- $\left(\sum_{p \in Space} v_p = 1\right) \in LC(\mathscr{S}, id, t)$,
- $\forall p \in Space \quad (v_p \geq 0) \in LC(\mathscr{S}, id, t)$
- No other constraints are in $LC(\mathscr{S}, id, t)$.

The problem of checking the consistency of a SPOT database was addressed in [50], where it was shown that SPOT database $\mathscr{S}$ is consistent iff $LC(\mathscr{S}, id, t)$ is feasible for all $\langle id, t \rangle$ pairs. The compatibility and entailment of a SPOT atom can be checked via the following result shown in [50].

Theorem 1. *Given a* SPOT *database* $\mathscr{S}$ *and a* SPOT *atom* $(id, r, t, [\ell, u])$,

i) $(id, r, t, [\ell, u]) \Subset \mathscr{S}$ *iff* $LC(\mathscr{S} \cup \{(id, r, t, [\ell, u])\}, id, t)$ *is feasible.*
ii) $\mathscr{S} \models (id, r, t, [\ell, u])$ *iff* $[\ell', u'] \subseteq [\ell, u]$ *where*
- $\ell' =$ ***minimize*** $\Sigma_{p \in r} v_p$ ***subject to*** $LC(\mathscr{S}, id, t)$
- $u' =$ ***maximize*** $\Sigma_{p \in r} v_p$ ***subject to*** $LC(\mathscr{S}, id, t)$.

Hence, the consistency of a SPOT database, as well as the compatibility and entailment of a SPOT atom, can be checked by a linear programming algorithm. The complexity of these algorithms was shown to be $O(|ID| \cdot |T| \cdot (|Space| \cdot |\mathscr{S}|)^3)$.

An optimized version of these algorithms was proposed in [49], where the number of variables of $LC(\cdot)$ was drastically reduced by introducing an equivalence relation on points in *Space*, giving a partition of *Space*, $\mathscr{P} = \{\mathscr{P}_1, \ldots, \mathscr{P}_n\}$ so that all the points in any $\mathscr{P}_i$ cannot be distinguished by $\mathscr{S}^{id,t}$. It was proved that consistency, compatibility, and entailment can be checked by considering a version of $LC(\cdot)$ (namely $PLC(\cdot)$) where the points in *Space* are replaced by the partitions in $\mathscr{P}$, so that the points $p \in \mathscr{P}_i$ are replaced by a new single variable $v_{\mathscr{P}_i}$. As the size of $\mathscr{P}$ is typically very much lower than the number of points in *Space*, the amount of time needed for solving $PLC(\cdot)$ is very much smaller than that needed for solving $LC(\cdot)$ [49] experimentally shows that reduced-size algorithms are much more efficient than those introduced in [50].

5 Query Processing in SPOT

The most investigated kind of query in SPOT database is selection.

Definition 8 (Selection query). A *selection query* is an expression of the form $(?id, q, ?t, [\ell, u])$ where q is a region of *Space*, not necessarily rectangular, $[\ell, u]$ is a probability interval, $?id$ is a variable ranging over ids in *ID*, and $?t$ is a variable ranging over time points in T.

Intuitively, a selection query says: "Find all objects *id* and times t such that the object *id* is inside the specified region q at time t with a probability in the $[\ell, u]$ interval." There are two semantics for interpreting this statement, leading to two types of selection queries.

Definition 9 (Optimistic/Cautious selection). Suppose $\mathscr{S}$ is a SPOT database and $(?id, q, ?t, [\ell, u])$ is a selection query.
The *optimistic answer* to $(?id, q, ?t, [\ell, u])$ is the set

$$\{\langle id, t\rangle \mid id \in ID \wedge t \in T \wedge (id, q, t, [\ell, u]) \Subset \mathscr{S}\}.$$

The *cautious answer* to $(?id, q, ?t, [\ell, u])$ is the set

$$\{\langle id, t\rangle \mid id \in ID \wedge t \in T \wedge \mathscr{S} \models (id, q, t, [\ell, u])\}.$$

Optimistic selection returns objects and time points that *may* be in the query region with probability in the specified interval, whereas cautious selection only returns those objects and time points that are *guaranteed* to be in that region with probability in that interval. Thus, the cautious answer is a subset of the optimistic one.

Example 5. Continuing our running example, one may be interested in knowing the ids and time points of profiles that were in the room where the fingerprint sensor is located, with probability greater than 0.75. This can be expressed by the selection query $(?id, q, ?t, [0.75, 1])$, where q is the rectangle defined by constraints $0 \leq x \leq 6$ and $4 \leq y \leq 8$ (this query region includes the whole area d, a portion of area b, and

some other points). The optimistic answer of this query is the set $\{< id_1, 1 >, < id_1, 2 >, < id_1, 3 >, < id_2, 1 >\}$, whereas the cautious answer only contains the pair $< id_1, 1 >$.

Optimistic and cautious selection can be computed by exploiting the results of Theorem 1. Specifically, given the selection query $Q = (?id, q, ?t, [\ell, u])$ over the SPOT database $\mathscr{S}$, the optimistic answer to Q can be computed by solving, for each pair $\langle id, t \rangle$ in $\mathscr{S}$, the linear program $LC(\mathscr{S} \cup \{(id, q, t, [\ell, u])\}, id, t)$: if it has a solution then $\langle id, t \rangle$ is in the optimistic answer of Q. The cautious answer to Q can be computed by solving, for each pair $\langle id, t \rangle$ in $\mathscr{S}$, the two optimization problems of Theorem 1(ii) which return the interval $[\ell', u']$. Then, checking if $[\ell', u'] \subseteq [\ell, u]$ is sufficient for deciding if the pair $\langle id, t \rangle$ is in the cautious answer of Q. We refer to these approaches as the *naive* algorithms for optimistic and cautious selection.

Efficient algorithms for computing optimistic and cautious selection were proposed in [49] and [47], respectively. Both proposed approaches exploit strategies for pruning the search space of candidate answers of a given query.

Optimistic Selection

An index structure, called SPOT -tree, and algorithms to compute optimistic answers to selection queries using the index were proposed in [49]. Each node of a SPOT -tree is labeled with a *composite* SPOT atom, which compactly represents a set of SPOT atoms (that is, a SPOT database). The relationship between parent and children nodes of a SPOT -tree is based on logical implication of SPOT databases. Basically, there is an entailment relationship between each composite atom labeling a child node and the composite atom labeling its parent node. Hence, the composite atom labeling the root node of a SPOT -tree is entailed by every composite atom labeling any node of the tree. Further, composite atoms labeling leaf nodes are entailed by SPOT atoms in the database.

The SPOT -tree reduces the set of $< id, t >$ pairs to be considered as candidates for the optimistic answer of a selection query — potentially, this set contains all the $< id, t >$ pairs in the SPOT database. The logical relationship between nodes of a SPOT -tree entails that, for each child node, the set of SPOT interpretations of its composite atom is a subset of the set of SPOT interpretations of its parent node composite atom. As a consequence, given a selection query Q, if the composite atom labeling the node n is not compatible with Q (this condition can be checked in constant time due to the structure of composite atoms), then any composite atom of the subtree rooted in n is not compatible with Q, which in turns means that all the $< id, t >$ pairs of SPOT -tree rooted in n do not belong to the optimistic answer of Q, i.e., they can be pruned from the search space.

For $< id, t >$ pairs that cannot be pruned by traversing the SPOT -tree, the naive algorithm is used for checking whether they are in the optimistic answer. Thus, for every pair pruned, we save the time needed for solving the linear programming problem of Theorem 1(i), as confirmed by the experimental study in [49].

Cautious Selection

In [47] the problem of efficiently computing cautious answers to selection queries was investigated. The proposed approach is based on geometric considerations which follow from the fact that both SPOT databases and selection queries define convex polytopes in the so-called SPOT PFD Space. The relationship between these polytopes can be used to answer queries. Specifically, it was shown that an $< id, t >$ pair belongs to the cautious answer of a given selection query iff the polytope defined by the query contains that defined by the SPOT database $\mathscr{S}^{id,t}$.

Approximations of the $\mathscr{S}^{id,t}$ polytope by interior and containing regions were introduced, instead of using the original $\mathscr{S}^{id,t}$ polytope which would have lead to an approach as computationally expensive as solving optimization problems of Theorem 1(ii). Thus, an $< id, t >$ pair is in the cautious answer of a given selection query if the query polytope contains a region containing the $\mathscr{S}^{id,t}$ polytope (this ensures that the query polytope contains that defined by $\mathscr{S}^{id,t}$). Similarly, an $< id, t >$ pair in *not* in the cautious answer (i.e., it can be pruned) if the query polytope does not contain an interior region of the $\mathscr{S}^{id,t}$ polytope. Both of these strategies can be jointly used to prune the search space when answering a query.

Efficient ways of finding interior and containing regions were also proposed. Containing regions can be obtained by starting from composite atoms introduced in [49] for defining SPOT -tree nodes. Internal regions can be obtained by following an inline or preprocessing approach. The former consist of storing solutions (that is, internal points of $\mathscr{S}^{id,t}$ polytope) of previously asked selection queries and then building the convex envelopes of found points. The latter approach, which is preferable to inlining when spare resources are available for precomputation, consists of solving a few optimization problems to find some internal points to be used to construct their convex envelope. As experimentally proved in [47], using either interior and containing regions yields improvement in terms of the efficiency of cautious selection.

SPOT *Query Algebra*

A relational-style algebra for SPOT databases has been proposed in [50]. This algebra contains a version of the selection, union, intersection, difference and join operators. Specifically, selection considered in [50] is a special case of cautious selection which returns ids only. Processing such queries by linear programming algorithms can be avoided if the database is known to satisfy the following *disjointness* property: there are no two distinct SPOT atoms in the database referring to the same object, the same time point and intersecting regions of *Space*.

Set operations and join in SPOT databases are more complicated and challenging than selection. We start by discussing the union operator, the simplest one. Union adds restrictions on the set of SPOT interpretations of the databases involved in the union operation. The intuition is that adding new information to a SPOT database reduces the degree of freedom that one has in choosing an interpretation. Formally, the union of SPOT databases $\mathscr{S}_1$ and $\mathscr{S}_2$ results in a SPOT database whose set

of interpretation is $\mathbf{I}(\mathscr{S}_1) \cap \mathbf{I}(\mathscr{S}_2)$. On the contrary, intersection and difference remove restrictions on SPOT interpretations of databases involved. Semantically, one would expect that $\mathbf{I}(\mathscr{S}_1 \cap \mathscr{S}_2) = \mathbf{I}(\mathscr{S}_1) \cup \mathbf{I}(\mathscr{S}_2)$ and $\mathbf{I}(\mathscr{S}_1 - \mathscr{S}_2) = \mathbf{I}(\mathscr{S}_1) \setminus \overline{\mathbf{I}}(\mathscr{S}_2)$, where $\overline{\mathbf{I}}(\mathscr{S}_2) = \{I \mid I \notin \mathbf{I}(\mathscr{S}_2)\}$. However, such semantics for intersection and difference cannot be expressed by SPOT databases since disjunction of probability intervals of SPOT atoms would be required. Hence, weaker definitions of intersection and difference operators returning supersets of the expected sets of interpretations (defined above) were introduced. Finally, a version of join was introduced that puts together information from different SPOT atoms having the same $< id, t >$ pair and satisfying an input function combining atom regions.

Nearest Neighbor Queries

Another kind of query in which SPOT database users may be interested is a nearest neighbor query, that is finding the nearest expected object neighbor to a given point of *Space* at a given time. As an example, a cell phone company may want to know the nearest cell phone to a given cell tower. Also the nearest neighbor to a given object, whose position is not known exactly, at a given time may be needed, such as finding the cell phone nearest to another one at a given time. Dealing with nearest neighbor queries implicitly requires dealing with the expected distances of objects from a given point or object at a given time. Here the expression "expected distance" is used in the strict sense of statistical expected values.

As SPOT databases do not state exactly where objects are at any given time, nor the exact probability that an object is in a given region of *Space* at a given time, minimal expected distance and maximal expected distance between objects were investigated in [50]. Then, these concepts were used to define two versions of nearest expected neighbor queries which return the object id whose expected distance from a given point at a given time is minimum or maximum, respectively. Under the assumption that the database is disjoint, algorithms for computing expected distance and nearest neighbor queries were proposed and experimentally validated.

6 Updates in SPOT

As SPOT databases provide information on moving objects, one of the aspect to be addressed is that information on these objects may continuously change as objects move. An object may encounter unexpected situations during its move, which may lead to a revision of estimates of where it may be in the future, as well as a revision of where it may have been in the past. The problem of revising SPOT data was first addressed in [48] and then further investigated in [27], where several strategies for revising SPOT data were proposed.

If the insertion of a SPOT atom a into a SPOT database $\mathscr{S}$ leads to no inconsistency then the atom can just be added to the database. However, when it leads to inconsistency, that is $\mathbf{I}(\mathscr{S} \cup \{a\}) = \emptyset$, then many different belief revision operations are possible. A first revision strategy proposed in [27] consists of finding a maximal

(w.r.t. either subset inclusion or cardinality) subset $\mathscr{S}'$ of $\mathscr{S}$ such that $\mathscr{S}' \cup \{a\}$ is consistent.

Example 6. Assume that the SPOT atom $a = (id_1, e, 3, [0.5, 1])$ providing new fresh information on the profile id_1 recognized at time 3 by the sensor in area e should be added to the SPOT database $\mathscr{S}_{lab}$ of Example 1. It is easy to see that $\mathscr{S}_{lab} \cup \{a\}$ is not consistent, as id_1 cannot stay at the same time in the disjoint areas c and e with probability in the intervals $[0.7, 0.8]$ and $[0.5, 1]$, respectively. A maximal consistent subset revision of $\mathscr{S}_{lab}$ is $\mathscr{S}'_{lab} = \mathscr{S}_{lab} \setminus \{(id_1, c, 3, [0.7, 0.8])\}$. As this revision removes only one atom from the database, it is maximal under both the subset inclusion and the subset cardinality criterion.

Maximal consistent subset revision works at the tuple level in the sense that the basic primitive to update the SPOT database is deletion of whole tuples. Strategies exploiting finer basic primitives such as updates of attribute values of tuples are also possible. For instance, in the example above, a revision may consist of changing the value of the temporal component of the tuple $(id_1, c, 3, [0.7, 0.8])$ in $\mathscr{S}_{lab}$ from 3 to 2. Revising strategies consisting of minimally modifying the spatial, temporal, or object components in $\mathscr{S}$ were investigated in [27]. Further, three revision mechanisms based on minimally revising the probability intervals in a SPOT database were addressed. In particular, by changing the relevant atoms' probability intervals to $[0, 1]$, consistency is restored; however, such a change loses much information and will usually not be minimal. It turns out that all of above-cited revision mechanisms lead to computational intractability except the strategy that revises the probability bounds. Probability revision can be computed by solving a linear programming problem similar to $LC(\mathscr{S}, id, t)$ (see Definition 7) where, for each SPOT atom a_i, additional variables low_i and up_i are used to represent the atom's modified lower and upper probability bounds.

Example 7. A probability revision for the SPOT database $\mathscr{S}_{lab}$ w.r.t. atom $a = (id_1, e, 3, [0.5, 1])$ is obtained by solving the following linear programming problem which is obtained by considering that $\mathscr{S}_{lab}^{id_1,3} = \{(id_1, b, 3, [0.6, 1]), (id_1, c, 3, [0.7, 0.8])\}$:

$$
\begin{array}{l}
\textbf{\textit{minimize}}\ (0.6 - low_1) + (up_1 - 1) + (0.7 - low_2) + (up_2 - 0.8) \\
\textbf{\textit{subject to}} \\
\left\{ \begin{array}{l}
low_1 \leq \sum_{p \in b} v_p \leq up_1 \\
0 \leq low_1 \leq 0.6 \\
1 \leq up_1 \leq 1 \\
low_2 \leq \sum_{p \in c} v_p \leq up_2 \\
0 \leq low_2 \leq 0.7 \\
0.8 \leq up_2 \leq 1 \\
0.5 \leq \sum_{p \in e} v_p \leq 1 \\
\sum_{p \in Space} v_p = 1 \\
\forall p \in Space \quad v_p \geq 0
\end{array} \right.
\end{array}
$$

Basically, for each atom $a_i = (id, r_i, t, [\ell_i, u_i])$ in the database, where id and t are the object identifier and the time point of the atom added to the database, the modified

probability interval $[low_i, up_i]$ is such that $[\ell_i, u_i] \subseteq [low_i, up_i] \subseteq [0,1]$. The inequalities with variables v_p, where p is a point in $Space$, ensure that there is a PDF satisfying the revision atom and every atom a_i in the database with $[\ell_i, u_i]$ replaced by $[low_i, up_i]$. Finally, the objective function guarantees that the revised probability bounds are as close as possible to the original ones.

In [48] and [27], the SPOT framework was extended to remove the assumptions that there are no velocity constraints on moving objects and that all points in $\mathscr{S}$ are reachable from all other points by all objects. A general notion of *reachability definition* was provided to capture the scenario where objects have speed limits and only some points are reachable by objects depending on both the distance between the points, the objects' speed, and possible obstacles in the way. Reachability constraints require the use of more complex variables in the linear programming problem of Definition 7. However, in this case the size of the problem increases quadratically in the size of $Space$ and the maximum time range in $\mathscr{S}$. As exact optimizations like that discussed after Theorem 1 do not work in this case due to the new structure of the linear programming problem, a suite of heuristics was proposed in [27] in order to speed up the computation of probability revision.

7 Research Questions about SPOT

As sketched in the previous sections we have found SPOT to be a very useful approach to the study of probabilistic spatiotemporal databases. We believe that there are interesting research problems within the SPOT framework as well through various extensions of SPOT . The purpose of this section is to specify and elaborate on such issues, thereby inviting researchers to do further work in this area. The topics are mostly independent of each other, although some could be combined.

We present five general topics. The first two are within the present SPOT framework; the other three present extensions to SPOT . First we discuss additional types of selection queries not considered in our papers. Then we suggest investigating the handling of aggregates and views. Recalling that SPOT databases use spatial regions and probability intervals, we suggest an extension to SPOT that allows for groups of objects and temporal intervals. Another possible extension involves adding connectives and quantifiers, thereby creating a full SPOT logic. Finally we suggest combining SPOT with logical terminology involving space and time.

7.1 Additional Selection Queries

As we discussed in Section 4, in our papers we dealt with selection queries of the form $(?id, q, ?t, [\ell, u])$ finding pairs of $\langle id, time \rangle$ values that provide answers to the query for a specific region q and probability interval $[\ell, u]$ using both the optimistic and cautious semantics. For the purpose of this section we will write such a query as $\{\langle id, t \rangle \mid (id, q_c, t, [\ell_c, u_c])\}$ indicating by the use of the subscript c that the region and probability values are fixed constants. We dealt with this selection query because we thought it was important, and we devised index structures for its efficient processing.

We believe that these and other index structures will be useful in answering other types of selection queries. In this subsection we list some queries with a similar form.

Let us start with atomic queries. Considering these systematically we start with one variable queries where the answer will be a set of values. Two of these appear interesting, namely $\{\langle id\rangle \mid (id, q_c, t_c, [\ell_c, u_c])\}$ and $\{\langle t\rangle \mid (id_c, q_c, t, [\ell_c, u_c])\}$. These are just like the queries we considered previously, but simpler, because in the first case we are considering a single time value and in the second case a single id value. The other two one variable queries appear less interesting because in the case of $\{\langle q\rangle \mid (id_c, q, t_c, [\ell_c, u_c])\}$ there may be many regions in the answer that are almost the same; while for $\{\langle [\ell, u]\rangle \mid (id_c, q_c, t_c, [\ell, u])\}$ the probability interval $[0, 1]$ is always in the answer. In the latter case we would likely be interested only in the smallest such probability interval. So, for example, if the answer is $[.8, 1]$, that means that id_c is in the region q_c at time t_c with probability at least .8.

We move on to two variable queries. Altogether there are 6 two variable queries. The query $\{\langle id, t\rangle \mid (id, q_c, t, [\ell_c, u_c])\}$ is the one we considered in the previous papers. As suggested for the one variable queries, asking for a region is less interesting, but asking for the smallest probability interval may be useful. For example, the query $\{\langle id, [\ell, u]\rangle \mid (id, q_c, t_c, [\ell, u])\}$ asks for all the objects and their smallest probability intervals such that the object is in region q_c at time t_c with that probability interval. Then, $\{\langle t, [\ell, u]\rangle \mid (id_c, q_c, t, [\ell, u])\}$ is interpreted in a similar way. Among the three variable queries probably the most interesting one is $\{\langle id, t, [\ell, u]\rangle \mid (id, q_c, t, [\ell, u])\}$ which essentially asks for all information about a region, that is, what objects were in the region at what times with what probability intervals.

Next we consider queries with connectives. Although it is not necessary to use the same selection queries that we used in the previous papers, we will do so here. Starting with conjunction, consider the query $\{\langle id, t\rangle \mid (id, q_c, t, [\ell_c, u_c]) \wedge (id, q_d, t, [\ell_d, u_d])\}$ asking for all $\langle id, t\rangle$ pairs such that the object was at that time both in region q_c with probability in the interval $[\ell_c, u_c]$ and in the region q_d with probability in the interval $[\ell_d, u_d]$. The result will be the intersection of two answers to two atomic queries. Similarly, the query $\{\langle id, t\rangle \mid (id, q_c, t, [\ell_c, u_c]) \vee (id, q_d, t, [\ell_d, u_d])\}$ gives the union of two atomic queries, and $\{\langle id, t\rangle \mid \neg(id, q_c, t, [\ell_c, u_c])\}$ gives the complement of the query answer without the negation but only if we also switch between optimistic and cautious selection. Finally, we add quantifiers. We consider only two examples. The query $\{\langle id\rangle \mid \exists t(id, q_c, t, [\ell_c, u_c])\}$ gives all objects that were at some time value in the region q_c with the probability in $[\ell_c, u_c]$. The same query with the universal quantifier, namely $\{\langle id\rangle \mid \forall t(id, q_c, t, [\ell_c, u_c])\}$ gives all objects that were in the region q_c with probability in $[\ell_c, u_c]$ for all time values.

7.2 Aggregates and Views

[57] is the definitive work on aggregates in probabilistic databases. A typical example in that paper concerns a day-ahead energy market with probabilities for various

prices. Basic aggregate queries involve finding the mean or standard deviation over prices for specified values of other attributes. In the SPOT framework this does not apply directly as we don't actually deal with numeric values (although we use integers for time values). Even so we may want to count the number of objects in a certain region at a certain time. Here we would compute the expected value based on the probabilities. The research problem is the application of the algorithms developed earlier for probabilistic aggregates in the SPOT framework.

Actually, we could add genuine numeric values to SPOT . Consider that there may be a benefit value given to the presence of an object being in a certain region at a certain time. This might be helpful, say, in a military operation or in the case of a taxi service. In this case we would augment the standard 4 columns by adding a fifth column of numeric values. We could then try to solve the problem of answering typical aggregate queries in an efficient way.

Views have been found very useful in relational databases. In the SPOT framework the most useful views would be obtained by asking a selection query that preserves all the attributes, so that the end result is also a SPOT database. Here the query $\{\langle id_c, q, t, [\ell, u]\rangle\}$ selects all the SPOT atoms for id_c. Thus one user may have a view with information only about a single object. Standard questions about views involve answering queries using views and view maintenance. In this simple example the view given can answer only queries about one object, id_c (it gives the null result for all other objects). View maintenance may involve atoms not in the view in case there are obstacles and speed issues concerning the movement of objects that in previous papers we modeled by using a reachability definition. The challenge here is to check when this needs to be done and find ways to speed up the process.

7.3 Groups of Objects and Time Intervals

In the SPOT framework some of the atomic values represent a set. In particular, a region is a set of points in *Space*; we have dealt mainly with rectangular regions, but only for ease of implementation. Also, for probabilities we allow probability intervals. Now we show how to extend SPOT for objects and time intervals.

In the case of objects we may be interested in a group of objects. For example, a military convoy may consist of multiple vehicles moving together. We can use the notation $(G, q, t, [\ell, u])$ to mean that the group of objects G is in region q at time t with probability in $[\ell, u]$. In such a case, if say $G = \{id_1, id_2, id_3\}$, we could have written this in SPOT as $(id_i, q, t, [\ell, u])$ for $i = 1, 2, 3$, that is, as three SPOT atoms. But there may be another meaning assigned to the atom $(G, q, t, [\ell, u])$: namely, that some object in G is in the region q at time t with probability interval $[\ell, u]$. The latter is the disjunctive interpretation of G, while the former is the conjunctive interpretation. The disjunctive interpretation cannot be expressed in SPOT .

In the case of time it would be natural to extend a single time value to a time interval. Consider the notation $(id, q, [t_1, t_2], [\ell, u])$ to indicate that the object id is in region q at times between t_1 and t_2 with probability in $[\ell, u]$. Again, there is a conjunctive and a disjunctive interpretation. In the conjunctive interpretation the

object is in the region with the given probability interval for all time values between t_1 and t_2, while in the disjunctive interpretation it is the case for some time value between t_1 and t_2.

The two cases, objects and times, can be combined to obtain the new generalized atom $(G,q,[t_1,t_2],[\ell,u])$ where there are four possible meanings using the conjunctive and disjunctive interpretations. The research challenge then is to extend the previous SPOT results to such generalized atoms. This involves answering various types of queries and doing database updates in the sense of revision. An interesting question is what data structures will be useful for speeding up processing in these cases.

7.4 *Full* SPOT *Logic*

The SPOT framework allows only atomic formulas, but these formulas have a strong expressive capability. Still, it is reasonable to consider what can be done if we allow connectives and quantifiers in the language. Actually, in such a case it will be useful also to allow both open and half-open probability intervals, so that, for example, $(id,q,t,(\ell,u])$ states that id is in region q at time t with probability interval greater than ℓ and at most u.

Starting with conjunction we note that a conjunction of SPOT atoms, $a_1 \wedge \ldots \wedge a_n$ is equivalent to the set $\{a_1,\ldots,a_n\}$. There is a difference however in updating a SPOT database. Recall that in order to retain consistency we introduced several strategies (see Section 5). Suppose that the conflict of a new atom is with a_1 only. Using maximal consistent set revision with $\{a_1,\ldots,a_n\}$ we delete a_1, but if we have the conjunction only, we must delete it, thereby losing the information in the other a_i, $i=2,\ldots,n$. Even if we apply a different strategy, such as minimizing probability, the definitions must be rewritten allowing, for example, in $a_1 \wedge \ldots \wedge a_n$ to change only one probability interval.

Disjunction presents substantial challenges (but then the same goes for going from relational to disjunctive databases). It is not clear that we can write a single linear program where the existence of a solution signifies the consistency of a disjunctive database, such as, if the disjunction $(id_1,q_1,t_1,[\ell_1,u_1]) \vee (id_2,q_2,t_2,[\ell_2,u_2])$ is included. Thus it may not be possible to adapt our approach using linear programming for this case. So it may be appropriate to limit disjunction in certain ways. For example, we may allow only one of the values to change, as in $(id_1,q,t,[\ell,u]) \vee (id_2,q,t,[\ell,u])$ where the identity of the object is known to be id_1 or id_2. Actually, this is the same situation that we considered in the previous subsection where we allowed groups of objects. However, the disjunction $(id,q,t,[\ell_1,u_1]) \vee (id,q,t,[\ell_2,u_2])$ assuming that $u_1 < \ell_2$ is not covered there.

Let's consider negation next. Here is where our extension to half-open intervals becomes useful. Essentially, the negation of an atom in SPOT can be expressed in the form of a disjunction, as follows: $\neg(id,q,t,[\ell,u]) \equiv (id,q,t,[0,\ell)) \vee (id,q,t,(u,1])$, with the appropriate modification in case $\ell = 0$ or $u = 1$. In that sense negation can be eliminated in full SPOT logic.

Finally, we take a look at quantifiers. In case *Id*, *Space* and *Time* are finite and we do not quantify over probabilities, we can rewrite quantified statements using conjunction and disjunction. For example, if $T = \{0,\ldots,N\}$, the statement $\exists t(id_c, q_c, t, [\ell_c, u_c])$, expressing that the object id_c was in region q_c at some time value with probability in the interval $[\ell_c, u_c]$ can be written without the existential quantifier as the disjunction $(id_c, q_c, 0, [\ell_c, u_c]) \vee \ldots \vee (id_c, q_c, N, [\ell_c, u_c])$. Similarly, changing the existential quantifier to a universal quantifier results in the disjunction changed to a conjunction. However, if we choose T to be infinite, then the quantifier cannot be replaced.

7.5 Adding Spatial and Temporal Concepts

Space and time are essential components in the SPOT framework. However, although SPOT is based on logical formalism, it does not take into account any concepts from the logics that have been developed for space and time. This subsection discusses some ideas that that could be used to augment the SPOT framework with spatial and temporal logic concepts.

We refer to the survey paper [32] for logical formalisms concerning spatial concepts. There are formal logical systems for spatial relations between regions such as *part_of* and *overlap* as well as spatial properties of regions such as *open*, *closed*, and *connected*, as well as operators such as *closure* and *boundary*. Thus we propose that this extension contain the usual SPOT atoms as well as atomic formulas that represent spatial concepts, and a proper mixture of the two. We assume here that regions are not restricted to rectangles in *Space*; in fact *Space* itself can be any set of points.

Allowing O to represent the overlap relation, the following three atoms: $(id, q, t, [\ell, u])$; $(id, r, t, [\ell, u])$; $O(q, r)$ clearly give more information than just the first two. We would also like to allow operators inside SPOT atoms; for example $(id, B(q), t, [\ell, u])$, where B stands for boundary, indicates the probability interval for the object *id* to be on the boundary of region q at time t. For connectedness, in the case of moving objects we may wish to define several types, such as a region is *car-connected* if one can travel by car from any point to any other point in the region versus *plane-connected* where travel is by plane. The research here involves the best spatial extensions and algorithms to deal with them.

Important issues about temporal logic are well summarized in [26]. As explained there, temporal logics have temporal operators such as F: "It will at some time be the case that ... " and G: "It will always be the case that ... ". Using such temporal operators to make general statements we may allow SPOT statements without the time component to write $G(id, q, [.8, 1])$ to mean that the object *id* will always be in the region q with probability at least .8. If time were finite, say $T = \{0, 1, \ldots, N\}$ and the present time *now* is known, then we could have written the $N - now$ statements $(id, q, n, [.8, 1])$ for $n = now + 1, \ldots, N$ with the same meaning. However, this cannot be done with our assumption that T is the set of integers. Also, even with finite time

we cannot express $F(id,q,[.8,1])$. So, in general, temporal operators expand the expressibility of the SPOT framework.

In the previous subsection we considered the addition of quantifiers to SPOT . Using quantifiers and inequality predicates we can express the temporal operators. For example, $F(id,q,[.8,1])$ would be written as $\exists t(t > now \wedge (id,q,t,[.8,1])$; G is the same but with the universal quantifier. However, quantifiers can cause complications and if our interest is only in expressing various temporal concepts, it may be better to use temporal operators. The important issue is the development of data structures to process such concepts as efficiently as possible. Finally, we mention the combination of spatial and temporal concepts that might be added to the SPOT framework for widening its expressibility.

References

1. Agarwal, P.K., Arge, L., Erickson, J.: Indexing moving points. Journal of Computer and System Sciences 66(1), 207–243 (2003)
2. Aziz, A., Sanwal, K., Singhal, V., Brayton, R.K.: Verifying continuous time markov chains. In: Alur, R., Henzinger, T.A. (eds.) CAV 1996. LNCS, vol. 1102, pp. 269–276. Springer, Heidelberg (1996)
3. Barbará, D., Garcia-Molina, H., Porter, D.: The management of probabilistic data. IEEE TKDE 4(5), 487–502 (1992)
4. Benjelloun, O., Sarma, A.D., Halevy, A.Y., Widom, J.: Uldbs: Databases with uncertainty and lineage. In: VLDB, pp. 953–964 (2006)
5. Bennett, B.: Modal logics for qualitative spatial reasoning. Journal of the Interest Group on Pure and Applied Logic 4, 23–45 (1996)
6. Brusoni, V., Console, L., Terenziani, P., Pernici, B.: Extending temporal relational databases to deal with imprecise and qualitative temporal information. In: Clifford, S., Tuzhilin, A. (eds.) Intl. Workshop on Recent Advances in Temporal Databases, pp. 3–22. Springer (1995)
7. Cao, H., Wolfson, O., Trajcevski, G.: Spatio-temporal data reduction with deterministic error bounds. VLDB Journal 15, 211–228 (2006)
8. Cavallo, R., Pittarelli, M.: The theory of probabilistic databases. In: VLDB, pp. 71–81 (1987)
9. Chen, Y.F., Qin, X.L., Liu, L.: Uncertain distance-based range queries over uncertain moving objects. J. Comput. Sci. Technol. 25(5), 982–998 (2010)
10. Chung, B.S.E., Lee, W.C., Chen, A.L.P.: Processing probabilistic spatio-temporal range queries over moving objects with uncertainty. In: EDBT, pp. 60–71 (2009)
11. Cohn, A.G., Hazarika, S.M.: Qualitative spatial representation and reasoning: an overview. Fundam. Inf. 46(1-2), 1–29 (2001)
12. Dai, X., Yiu, M.L., Mamoulis, N., Tao, Y., Vaitis, M.: Probabilistic spatial queries on existentially uncertain data. In: Medeiros, C.B., Egenhofer, M., Bertino, E. (eds.) SSTD 2005. LNCS, vol. 3633, pp. 400–417. Springer, Heidelberg (2005)
13. Dalvi, N.N., Suciu, D.: Efficient query evaluation on probabilistic databases. VLDB J 16(4), 523–544 (2007)
14. Dekhtyar, A., Dekhtyar, M.I.: Possible worlds semantics for probabilistic logic programs. In: Demoen, B., Lifschitz, V. (eds.) ICLP 2004. LNCS, vol. 3132, pp. 137–148. Springer, Heidelberg (2004)

15. Dey, D., Sarkar, S.: A probabilistic relational model and algebra. ACM Trans. Database Syst. 21(3), 339–369 (1996)
16. Dubois, D., Prade, H.: Processing fuzzy temporal knowledge. IEEE Transactions on Systems, Man and Cybernetics 19(4), 729–744 (1989)
17. Dutta, S.: Generalized events in temporal databases. In: Proc. 5th Intl. Conf. on Data Engineering, pp. 118–126 (1989)
18. Eiter, T., Lukasiewicz, T., Walter, M.: A data model and algebra for probabilistic complex values. Ann. Math. Artif. Intell. 33(2-4), 205–252 (2001)
19. Fagin, R., Halpern, J.Y., Megiddo, N.: A logic for reasoning about probabilities. Inf. Comput. 87(1/2), 78–128 (1990)
20. Fusun Yaman, D.N., Subrahmanian, V.: The logic of motion. In: Proc. 9th International Conference on the Principles of Knowledge Representation and Reasoning (KR 2004), pp. 85–94 (2004)
21. Fusun Yaman, D.N., Subrahmanian, V.: Going far, logically. In: Proc. IJCAI 2005, pp. 615–620 (2005)
22. Fusun Yaman, D.N., Subrahmanian, V.: A motion closed world assumption. In: Proc. IJCAI 2005, pp. 621–626 (2005)
23. Cohn, A.G., Magee, D.R., Galata, A., Hogg, D.C., Hazarika, S.M.: Towards an architecture for cognitive vision using qualitative spatio-temporal representations and abduction. In: Freksa, C., Brauer, W., Habel, C., Wender, K.F. (eds.) Spatial Cognition III. LNCS (LNAI), vol. 2685, pp. 232–248. Springer, Heidelberg (2003)
24. Gabelaia, D., Kontchakov, R., Kurucz, A., Wolter, F., Zakharyaschev, M.: On the computational complexity of spatio-temporal logics. In: FLAIRS Conference, pp. 460–464 (2003)
25. Gadia, S., Nair, S., Poon, Y.: Incomplete infromation in relational temporal databases. In: Proc. Intl. Conf. on Very Large Databases (1992)
26. Galton, A.: Temporal logic. In: Zalta, E.N. (ed.) The Stanford Encyclopedia of Philosophy, fall 2008 edn. (2008)
27. Grant, J., Parisi, F., Parker, A., Subrahmanian, V.S.: An agm-style belief revision mechanism for probabilistic spatio-temporal logics. Artif. Intell. 174(1), 72–104 (2010)
28. Güntzer, U., Kiessling, W., Thöne, H.: New direction for uncertainty reasoning in deductive databases. In: Proceedings of the 1991 ACM SIGMOD International Conference on Management of Data, SIGMOD 1991, pp. 178–187. ACM, New York (1991), http://doi.acm.org/10.1145/115790.115815
29. Hadjieleftheriou, M., Kollios, G., Tsotras, V.J., Gunopulos, D.: Efficient indexing of spatiotemporal objects. In: Jensen, C.S., Jeffery, K., Pokorný, J., Šaltenis, S., Bertino, E., Böhm, K., Jarke, M. (eds.) EDBT 2002. LNCS, vol. 2287, pp. 251–268. Springer, Heidelberg (2002)
30. Hansson, H., Jonsson, B.: Logic for reasoning about time and reliability. Formal Asp. Comput. 6(5), 512–535 (1994)
31. Jampani, R., Xu, F., Wu, M., Perez, L.L., Jermaine, C., Haas, P.J.: MCDB: a monte carlo approach to managing uncertain data. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 687–700. ACM, New York (2008)
32. Jeansoulin, R., Papini, O., Prade, H., Schockaert, S.: Introduction: uncertainty issues in spatial information. In: Jeansoulin, R., Papini, O., Prade, H., Schockaert, S. (eds.) Methods for Handling Imperfect Spatial Information. STUDFUZZ, vol. 256, pp. 1–14. Springer, Heidelberg (2010)
33. Kifer, M., Li, A.: On the semantics of rule-based expert systems with uncertainty. In: ICDT, pp. 102–117 (1988)

34. Koch, C., Olteanu, D.: Conditioning probabilistic databases. Proceedings of the VLDB Endowment archive 1(1), 313–325 (2008)
35. Kollios, G., Gunopulos, D., Tsotras, V.J.: On indexing mobile objects. In: Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp. 261–272. ACM, New York (1999)
36. Koubarakis, M.: Database models for infinite and indefinite temporal information. Information Systems 19(2), 141–173 (1994)
37. Kwiatkowska, M., Norman, G., Parker, D.: PRISM: Probabilistic symbolic model checker. In: Field, T., Harrison, P.G., Bradley, J., Harder, U. (eds.) TOOLS 2002. LNCS, vol. 2324, pp. 200–204. Springer, Heidelberg (2002)
38. Lakshmanan, L.V., Sadri, F.: Modeling uncertainty in deductive databases. In: Karagiannis, D. (ed.) DEXA 1994. LNCS, vol. 856, pp. 724–733. Springer, Heidelberg (1994)
39. Lakshmanan, L.V.S., Shiri, N.: A parametric approach to deductive databases with uncertainty. IEEE Trans. on Knowl. and Data Eng. 13(4), 554–570 (2001)
40. Lian, X., Chen, L.: Probabilistic group nearest neighbor queries in uncertain databases. IEEE Transactions on Knowledge and Data Engineering 20(6), 809–824 (2008), http://doi.ieeecomputersociety.org/10.1109/TKDE.2008.41
41. Lukasiewicz, T.: Probabilistic logic programming. In: ECAI, pp. 388–392 (1998)
42. Lukasiewicz, T., Kern-Isberner, G.: Probabilistic logic programming under maximum entropy. In: Hunter, A., Parsons, S. (eds.) ECSQARU 1999. LNCS (LNAI), vol. 1638, pp. 279–292. Springer, Heidelberg (1999)
43. Merz, S., Wirsing, M., Zappe, J.: A spatio-temporal logic for the specification and refinement of mobile systems. In: Pezzé, M. (ed.) FASE 2003. LNCS, vol. 2621, pp. 87–101. Springer, Heidelberg (2003)
44. Muller, P.: Space-Time as a Primitive for Space and Motion. In: FOIS, pp. 63–76. IOS Press, Amsterdam (1998), http://www.irit.fr/~Philippe.Muller
45. Ng, R.T., Subrahmanian, V.S.: Probabilistic logic programming. Information and Computation 101(2), 150–201 (1992), citeseer.csail.mit.edu/ng92probabilistic.html
46. Ni, J., Ravishankar, C.V., Bhanu, B.: Probabilistic spatial database operations. In: Hadzilacos, T., Manolopoulos, Y., Roddick, J., Theodoridis, Y. (eds.) SSTD 2003. LNCS, vol. 2750, pp. 140–159. Springer, Heidelberg (2003)
47. Parisi, F., Parker, A., Grant, J., Subrahmanian, V.S.: Scaling cautious selection in spatial probabilistic temporal databases. In: Jeansoulin, R., Papini, O., Prade, H., Schockaert, S. (eds.) Methods for Handling Imperfect Spatial Information. STUDFUZZ, vol. 256, pp. 307–340. Springer, Heidelberg (2010)
48. Parker, A., Infantes, G., Grant, J., Subrahmanian, V.: An agm-based belief revision mechanism for probabilistic spatio-temporal logics. In: AAAI (2008)
49. Parker, A., Infantes, G., Grant, J., Subrahmanian, V.S.: Spot databases: Efficient consistency checking and optimistic selection in probabilistic spatial databases. IEEE TKDE 21(1), 92–107 (2009)
50. Parker, A., Subrahmanian, V.S., Grant, J.: A logical formulation of probabilistic spatial databases. IEEE TKDE, 1541–1556 (2007)
51. Parker, A., Yaman, F., Nau, D., Subrahmanian, V.: Probabilistic go theories. In: IJCAI, pp. 501–506 (2007)
52. Pelanis, M., Saltenis, S., Jensen, C.S.: Indexing the past, present, and anticipated future positions of moving objects. ACM Trans. Database Syst. 31(1), 255–298 (2006)
53. Pfoser, D., Jensen, C.S., Theodoridis, Y.: Novel approaches to the indexing of moving object trajectories. In: Proceedings of VLDB (2000)
54. Pittarelli, M.: An algebra for probabilistic databases. IEEE TKDE 6(2), 293–303 (1994)

55. Rajagopalan, R., Kuipers, B.: Qualitative spatial reasoning about objects in motion: Application to physics problem solving. In: IJCAI 1994, San Antonio, TX, pp. 238–245 (1994)
56. Randell, D.A., Cui, Z., Cohn, A.G.: A spatial logic based on regions and connection. In: International Conference on Knowledge Representation and Reasoning, KR 1992, pp. 165–176. Morgan Kaufmann (1992)
57. Ross, R., Subrahmanian, V.S., Grant, J.: Aggregate operators in probabilistic databases. Journal of the ACM 52(1), 54–101 (2005)
58. Shanahan, M.: Default reasoning about spatial occupancy. Artif. Intell. 74(1), 147–163 (1995), http://dx.doi.org/10.1016/0004-3702(94)00071-8
59. Snodgrass, R.: The temporal query language tquel. In: Proceedings of the 3rd ACM SIGACT-SIGMOD Symposium on Principles of Database Systems, PODS 1984, pp. 204–213. ACM, New York (1984), http://doi.acm.org/10.1145/588011.588041
60. Tao, Y., Cheng, R., Xiao, X., Ngai, W.K., Kao, B., Prabhakar, S.: Indexing multi-dimensional uncertain data with arbitrary probability density functions. In: VLDB, pp. 922–933 (2005)
61. Tao, Y., Papadias, D., Sun, J.: The TPR*-tree: an optimized spatio-temporal access method for predictive queries. In: Proceedings of the 29th International Conference on Very Large Data Bases, vol. 29, pp. 790–801. VLDB Endowment (2003)
62. Wolter, F., Zakharyaschev, M.: Spatial reasoning in rcc-8 with boolean region terms. In: Principles of Knowledge Representation and Reasoning, ECAI 2000, pp. 244–248. IOS Press, Berlin (2000)
63. Wolter, F., Zakharyaschev, M.: Spatio-temporal representation and reasoning based on RCC-8. In: Cohn, A.G., Giunchiglia, F., Selman, B. (eds.) Principles of Knowledge Representation and Reasoning, KR 2000, pp. 3–14. Morgan Kaufmann, San Francisco (2000), citeseer.ist.psu.edu/wolter00spatiotemporal.html
64. Yang, B., Lu, H., Jensen, C.S.: Probabilistic threshold k nearest neighbor queries over moving objects in symbolic indoor space. In: Manolescu, I., Spaccapietra, S., Teubner, J., Kitsuregawa, M., Léger, A., Naumann, F., Ailamaki, A., Özcan, F. (eds.) EDBT. ACM International Conference Proceeding Series, vol. 426, pp. 335–346. ACM (2010)
65. Zhang, M., Chen, S., Jensen, C.S., Ooi, B.C., Zhang, Z.: Effectively indexing uncertain moving objects for predictive queries. PVLDB 2(1), 1198–1209 (2009)
66. Zheng, K., Trajcevski, G., Zhou, X., Scheuermann, P.: Probabilistic range queries for uncertain trajectories on road networks. In: Ailamaki, A., Amer-Yahia, S., Patel, J.M., Risch, T., Senellart, P., Stoyanovich, J. (eds.) EDBT, pp. 283–294. ACM (2011)