Semistructured Probabilistic Databases

Alex Dekhtyar
dekhtyar@cs.uky.edu
Department of Computer Science
University of Kentucky

Judy Goldsmith
goldsmit@cs.uky.edu
Department of Computer Science
University of Kentucky

Sean R. Hawkes
srhawk0@cs.uky.edu
Department of Computer Science
University of Kentucky

Abstract

This work describes a new theoretical framework for uniform storage and management of diverse probabilistic information.

1 Introduction

The need to store and manage *probabilistic information* can appear in a number of different applications, from multimedia databases storing the results of image recognition to logistics databases to stock market prediction software to a wide variety of applications of Bayesian Nets [17]. Over the past 13 years there have been a number of relational [6, 2, 9, 15] and object [14, 10] data models proposed to support storage and querying of probabilistic information. Unfortunately, these approaches are not sufficiently flexible to handle the different contexts in which probabilities must be discussed in analyzing a stochastic system. For instance, consider academic advising, where the expectation of a student's success may be represented in a variety of forms: a simple probability distribution for one course or a joint probability distribution for several courses, or a simple or joint conditional probability distribution (success in later courses may depend on earlier grades).

This variety of formats would require separate storage in any of the current probabilistic relational models, making even simple queries hard to express. Thus, we propose a new, *semistructured* probabilistic data model designed to alleviate this problem.

Semistructured data model [1, 5, 18] has gained wide acceptance recently as the means of representing the data which lacks a rigid structure of schema. In particular, the similarity of the semistructured data model and the underlying data model for Extensible Markup Language (XML) [4], the emerging open standard for data storage and transmission over the Internet make our choice of this approach attractive. In this paper, we present the formal model for semistructured probabilistic objects. This paper provides the theoretical foundations for storing and managing semistructured probabilistic objects. In [13] we have started the process of translating this model into XML.

In Section 2, we introduce the advising application. Section 3 gives formal definitions of semistructured probabilistic objects, and Section 4 introduces the underlying algebra for semistructured probabilistic databases.

2 Motivating Example

The following comes from our work on modeling academic advising as a process of uncertain inference [7].

Consider a database, designed to assist faculty members with the academic advising process. Typically, an advisor sees each advisee once every semester to suggest the set of courses to take next semester. The advisor tries to suggest courses that fulfill degree requirements and for which the student has the highest chance (probability) of success.

Statistical information about student performance in different classes can be extracted from the student transcript database maintained by every university. Under the assumption that this information correctly reflects (approximates) the true probabilities, it can be used to assist the advisor in her recommendation.

Note that the type of probabilistic information available to the advisor in this example varies greatly. The simplest is a *probability distribution of student performance in one course*. The advisor can compare the probability distribution for Databases with the probability distribution for

DB	P
A	0.2
B	0.3
C	0.4
F	0.1

DB	OS	P
A	A	0.05
A	B	0.05
A	C	0.1
A	F	0.01
B	A	0.04
B	B	0.08
...	...	...
F	C	0.02
F	F	0.03

major :	CS
college:	ENG
instructor:	Smith
instructor:	Jones

DB	OS	P
A	A	0.01
A	B	0.02
A	C	0.2
A	F	0.01
B	A	0.05
B	B	0.12
...	...	...
F	C	0.03
F	F	0.01

major :	CS
year :	2000

DB	OS	P
A	A	0.1
A	B	0.1
A	C	0.01
A	F	0
B	A	0.1
B	B	0.2
...	...	...
F	C	0.01
F	F	0

DataStructures= A
Logic $\in$ { A, B }

Figure 1. Different types of probabilistic information to be stored in the Advisor **database (from left to right: single variable probability distribution, joint probability distribution of 2 variables, joint probability distribution with context, conditional joint probability distribution with context.)**

Operating Systems in order to choose the course that has a higher probability of success.

Another type of probabilistic information that can be useful in this situation is a *joint probability distribution*. When the advisor needs to consider the entire list of courses for the student to take next semester, she is interested in the student's success in *all* courses at once. This brings up another type of probabilistic information that can be useful, a *joint probability distribution*.

To make matters more complicated, we notice that student's success in the future classes can depend intrinsically on her current grades. A C in a Data Structures class may suggest to the advisor that the student might not do well in Algorithms, while and A in Logic suggests a good chance of success in Artificial Intelligence. Other possible information that can affect the probability distribution may include some general information about the student/course background such as student's major, college, graduate/undergraduate status, professor who teaches the course, etc...

The possible types of probabilistic information to be stored a database for advising support are shown in Figure 1. There, and in the examples throughout the paper we limit possible grades to A, B, C, F.

When trying to store this data using one of the previously proposed probabilistic database models, relational or object, a number of problems will be encountered.

Probabilistic relational models [2, 9, 15] lack flexibility to store some of our data in a straightforward manner. In the advising application the courses are viewed as *random variables*. As such, it is natural to represent each course as a database attribute that can take values from the set {A,B,C,F}. However, with such interpretation, a joint probability distribution of student grades in two courses will have a schema different from the joint probability distribution of three courses, and therefore, will have to be stored in a separate relation. In such a database, expressing queries like "Find all probability distributions that include Databases as a random variable" is very inconvenient, if at all possible.

Probabilistic Object models [14, 10] are also not a good fit for storing this kind of data. In the framework of Eiter et al. [10], a probabilistic object is a "real" object, some of whose properties are uncertain and probabilistically described. For our application, the *probability distribution* is the object that needs to be stored.

With this example in mind, we proceed to describe our data model[1].

3 Data Model

Consider a universe $\mathcal{V}$ of *random variables* $\{v'_1, \ldots, v'_m\}$. With each random variable $v \in \mathcal{V}$ we associate $dom(v)$, the set of its *possible values*. Given a set $V = \{v_1, \ldots v_q\} \subseteq \mathcal{V}$, $dom(V)$ will denote $dom(v_1) \times \ldots dom(v_q)$.

Let $\mathcal{R} = (A_1, \ldots, A_n)$ be a collection of *regular relational attributes*. For $A \in \mathcal{R}$, $dom(A)$ will denote the domain of A. We define a *semistructured schema* $R^\star$ over $\mathcal{R}$ as a **multiset** of attributes from $\mathcal{R}$. For example, if $\mathcal{R} = \{\mathsf{year}, \mathsf{major}, \mathsf{college}\}$, the following are valid semistructured schemas over $\mathcal{R}$: $R_1^\star = \{\mathsf{year}, \mathsf{college}\}$; $R_2^\star = \{\mathsf{year}, \mathsf{year}, \mathsf{major}, \mathsf{college}\}$; $R_3^\star = \{\mathsf{major}, \mathsf{major}, \mathsf{major}\}$.

Let $\mathcal{P}$ denote a *probability space* used in the framework to represent probabilities of different events. Examples of

[1] A good advisor does not inform a student that she has probability x of getting grade y in a given course. Thus, all probabilities used in examples here were chosen randomly, and do not represent actual observed or computed probabilities for these events.

DB	OS	P
A	A	0.09
A	B	0.12
A	C	0.03
A	F	0.005

DB	OS	P
B	A	0.12
B	B	0.16
B	C	0.13
B	F	0.01

DB	OS	P
C	A	0.03
C	B	0.08
C	C	0.11
C	F	0.045

DB	OS	P
F	A	0
F	B	0.01
F	C	0.02
F	F	0.04

Table 1. Joint Probability Distribution of Student Performance in Databases and Operating Systems Courses.

such probability spaces include (but are not limited to) the interval $[0, 1]$ and the set C[0,1] of *all subintervals of* $[0, 1]$ [16, 8, 15]. For each probability space $\mathcal{P}$ there should exist a notion of a *consistent probability distribution over* $\mathcal{P}$[2].

We are ready to define the key notion of our framework: *Semistructured Probabilistic Objects (SPOs).*

Definition 1 *A* ***Semistructured Probabilistic Object (SPO)*** *S is defined as a quadruple $S = \langle T, V, P, C\rangle$, where*

- *T is a relational tuple over some* semistructured schema *$R^\star$ over $\mathcal{R}$. We will refer to T as the* **context** *of S.*
- *$V = \{v_1, \ldots, v_q\} \subseteq \mathcal{V}$ is a set of random variables that* **participate** *in S. We require that $V \neq \emptyset$.*
- *$P : dom(V) \longrightarrow \mathcal{P}$ is the* **probability table** *of S. Note that P need not be complete, but it* ***must*** *be* consistent *w.r.t. $\mathcal{P}$.*
- *$C = \{(u_1, X_1), \ldots (u_s, X_s)\}$, where $\{u_1, \ldots, u_s\} = U \subseteq \mathcal{V}$ and $X_i \subseteq dom(u_i)$, $1 \leq i \leq n$, such that $V \cap U = \emptyset$. We refer to C as the* **set of conditionals** *of S.*

An explanation of this definition is in order. In order for our data model to possess the ability to store all the probability distributions mentioned in Section 2 (see Figure 1), the following information needs to be stored in a single object:

1. **Participating random variables.** These variables determine the probability distribution described in an SPO.
2. **Probability Table.** If *only one* random variable participates, it is a *simple* probability distribution table; otherwise the distribution will be *joint*. Probability table may be *complete*, when the information about the probability of *every* instance is supplied, or *incomplete*.

   It is convenient to visualize the probability table P as a table of rows of the form $(\bar{x}, \alpha)$, where $\bar{x} \in dom(V)$ and $\alpha = P(\bar{x})$. Thus, we will speak about *rows* and *columns* of the probability table where it makes explanations more convenient.
3. **Conditionals.** A probability table may represent *conditional distribution*, conditioned by some prior information. The **conditional part** of its SPO stores the prior information in one of two forms: *"random variable u has value x"* or *"the value of random variable u is restricted to a subset X of its values"*. In our definition, this is represented as a pair (u, X). When X is a singleton set, we get the first type of the condition.
4. **Context** provides supporting information for a probability distribution – information about the known values of certain parameters, which **are not considered to be random variables** by the application.

Example 1 *Consider the joint probability distribution of student grades in* Databases (DB) *and* Operating Systems (OS) *for College of Engineering majors who have a grade of* A *or* B *in* Data Structures (DS) *defined in Table 1.*

We can break this information into our four categories as follows:

participating random variables: $V = \{\mathsf{DB}, \mathsf{OS}\}$.

probability table: *specified in Table 1. Here, the probability table defines a complete and consistent probability distribution.*

conditionals: *there is a single conditional* $\mathsf{DS} \in \{\mathsf{A}, \mathsf{B}\}$ *associated with this distribution, which is stored in an SPO as* $C = \{(\mathsf{DS}, \{\mathsf{A}, \mathsf{B}\})\}$.

context: *information about student's college within the University is* ***not*** *represented by a random variable in our universe. It is, therefore, represented as a relational attribute* college. *Thus,* college: Engineering *is the* ***context*** *of the probabilistic information in this example.*

[2] For $\mathcal{P} = [0, 1]$ the consistency constraint states that the sum of probabilities in a *complete* probability distribution must add up to exactly 1.

4 Semistructured Probabilistic Algebra

Let us fix the universe of random variables $\mathcal{V}$, the universe of context attributes $\mathcal{R}$, and the probability space $\mathcal{P}$. In the remainder of this paper we will assume that $\mathcal{P} = [0, 1]$.

A finite collection $\mathcal{S} = \{S_1, \ldots, S_n\}$ of *semistructured probabilistic objects* over $\langle \mathcal{V}, \mathcal{R}, \mathcal{P} \rangle$ is called a ***semistructured probabilistic relation (SPR)***. A finite collection $\mathcal{D}_S = \{\mathcal{S}_1, \ldots, \mathcal{S}_m\}$ is called a ***semistructured probabilistic database (SPD)***.

One important difference between *semistructured probabilistic databases* and classic relational or relational probabilistic databases is that different semistructured probabilistic relations have the same "schema" as their attributes range over the same universe $\langle \mathcal{V}, \mathcal{R}, \mathcal{P} \rangle$. Unlike relational databases, where tuples over different schemas must belong to different relations, any two SPOs can be in the same semistructured probabilistic relation. Thus, the division of a semistructured probabilistic database into relations is done for the purpose of specifying the structure of the database. For example, if the SPD is built from the information supplied by three different experts, this information can be arranged into three SPRs - according to the origin of each object inserted in the database.

The key to the efficient use of semistructured probabilistic databases in representing probabilistic information is the management of the data stored in SPDs. Just as with *probabilistic relational databases*, where probabilistic relational algebras of Barbara et al. [2], Dey and Sarkar [9] and Lakshmanan et al. [15] extend classical relational algebra by adding probability-specific (and probability theory compliant) manipulation of the probabilistic attributes in the relations, a new semistructured algebra needs to be developed for SPDs, in order to capture properly the manipulation of probabilities.

In the remainder of this section we introduce such algebra, called *Semistructured Probabilistic Algebra (SP-Algebra)*. This algebra will extend the definitions of standard relational operations select, project, cartesian product,join to account for the appropriate maintenance of probabilistic information within SPOs, as well as introduce new a operation, conditionalization (see also [9]), which is specific to the probabilistic nature of the data.

4.1 Selection

Given an SPO $S = \langle T, V, P, C \rangle$, selection operation may apply to any of its four parts. Each part requires its own language of selection constraints.

Selection on *context*, *participants* and *conditionals* will result in the entire SPO either being selected or not (as is the case with selection on classical relations). Selection on *probability table* it will result in only *part* of the original probability table P returned. A selection based on the probabilistic values is also possible and it should also result in only part of the probabilistic table returned. The five different types of selections are illustrated in the following example.

Example 2 *Consider the SPO S described in Example 1. The first three queries below return S.*

1. "Find all probability distributions related to College of Engineering majors".

2. "Find all probability distributions that involve the Operating Systems course".

3. "Find all probability distributions related to students who took Data Structures and got a grade of C or better".

4. "What information is available about the probability of getting an A in Databases?" Databases $S.P$ contains four entries that relate to the probability of getting an A in Databases. This part of $S.P$ should be returned as a result together with the T, V and C parts of S. The remainder of the $S.P$ should **not** be returned.

5. "What outcomes have the probability over 0.1?" In the probability table of S, there are five possible outcomes that have the probability greater than 0.1. In the result of executing this query on S, $S.P$ should contain exactly these five lines, with $S.T$, $S.V$ and $S.C$ remaining unchanged.

4.1.1 Selection on Context, Participation or Conditionals

Here, we define the three selection operations that do not alter the content of the selected objects. We start by defining the acceptable languages for selection conditions for the three types of selects.

Recall that the universe $\mathcal{R}$ of context attributes consists of a finite set of attributes $A_1, \ldots A_n$ with domains $dom(A_1), \ldots, dom(A_n)$. With each attribute $A \in \mathcal{R}$ we associate a set $Pr(A)$ of **allowed predicates**. We assume that equality and inequality are **allowed** for **all** $A \in \mathcal{R}$.

Definition 2 *1. An **atomic context selection condition** is an expression c of the form $A\ Q\ x$ ($Q(A, x)$), where $A \in \mathcal{R}$, $x \in dom(A)$ and $Q \in Pr(A)$.*

*2. An **atomic participation selection condition** is an expression c of the form $v \in V$, where $v \in \mathcal{V}$ is a random variable.*

S:

major: CS

DB	OS	P
A	A	0.1
A	B	0.1
B	A	0.13
B	C	0.09
C	C	0.16

Logic = A

$\sigma_{\mathrm{DB=A}}$**(S):**

major: CS

DB	OS	P
A	A	0.1
A	B	0.1

Logic = A

$\sigma_{\mathrm{P}>0.11}$**(S):**

major: CS

DB	OS	P
B	A	0.13
C	C	0.16

Logic = A

Figure 2. Selection on Probabilistic Table and on Probability values in SP-Algebra

3. *An* ***atomic conditional selection condition*** *is one of the following expressions:* $u = \{x_1, \dots x_h\}$ *or* $u \ni x$ *where* $u \in \mathcal{V}$ *is a random variable and* $x, x_1, \dots, x_h \in dom(u)$.

Complex selection conditions can be formed as Boolean combinations of atomic selection conditions.

Definition 3 *Let* $S = \langle T, V, P, C\rangle$ *is an SPO and let* $c : Q(A, x)$ *be an atomic context selection condition. Then* $\sigma_c(S) = \{S\}$ ***iff***

- $A \in R^\star$*;*
- *For some instance* $A^\star$ *of* A *in* T*,* $(S.T.A^\star, x) \in Q$*;*

otherwise $\sigma_c(S) = \emptyset$.

Definition 4 *Let* $S = \langle T, V, P, C\rangle$ *is an SPO and let* $c : v \in V$ *be an atomic participation selection condition. Then* $\sigma_c(S) = \{S\}$ ***iff*** $v \in V$.

Definition 5

1. *Let* $S = \langle T, V, P, C\rangle$ *is an SPO and let* $c : u = \{x_1, \dots, x_h\}$ *be an atomic conditional selection condition. Then* $\sigma_c(S) = \{S\}$ ***iff*** $C \ni (u, X)$ *and* $X = \{x_1, \dots, x_h\}$.
2. *Let* $c : u \ni x$ *be an atomic conditional selection condition. Then* $\sigma_c(S) = \{S\}$ ***iff*** $C \ni \langle (u, X)$ *and* $X \ni x$.

The semantics of atomic selection conditions can be extended to their Boolean combinations in a straightforward manner: $\sigma_{C \wedge C'}(S) = \sigma_C(\sigma_{C'}(S))$ and $\sigma_{C \vee C'}(S) = \sigma_C(S) \vee \sigma_{C'}(S)$.

The interpretation of *negation* in the context selection condition requires some additional explanation. In order for a selection condition of a form $\neg Q(A, x)$ to succeed on some SPO $S = \langle T, V, P, C\rangle$, attribute A **must be present** R_T. If A is not in $R^\star$, the selection condition does not get evaluated and the result will be $\emptyset$. Therefore, the statement $S \in \sigma_c(S) \vee S \in \sigma_{\neg c}(S)$ is not necessarily true. This also applies to conditional selection conditions.

Finally, $\sigma_C(\mathcal{S}) = \bigcup_{S \in \mathcal{S}}(\sigma_C(S))$.

4.1.2 Selection on Probability Table or Probabilities

Selection operations considered in previous section were simple in that their result on a semistructured probabilistic relation was always a subset of the relation.

The two types of selections introduced here are more complex. The result of each operation applied to an SPO can be *a non-empty* part of the original SPO. In particular, both operations preserve the context, participating random variables and conditionals in an SPO, but may return *only a subset* of the rows of the probability table. Both in the case of the selection on probability table and selection on probabilities, the selection condition will indicate which rows are to be included and which are to be omitted.

Definition 6 *An* atomic probabilistic table selection condition *is an expression of the form* $v = x$ *where* $v \in \mathcal{V}$ *and* $x \in dom(v)$. Probabilistic table selection conditions *are Boolean combinations of* atomic probabilistic table selection conditions.

Definition 7 *Let* $S = \langle T, V, P, C\rangle$ *be an SPO,* $V = \{v_1, \dots, v_k\}$ *and let* $c : v = x$ *be an atomic probabilistic table selection condition.*

If $v \in V$*, then (assume* $v = v_i, 1 \le i \le k$*), the result of selection from* S *on* c*,* $\sigma_c(S)$ *is a semistructured probabilistic object* $S' = \langle T, V, P', C\rangle$*, where*

$$P'(y_1, \dots, \mathbf{y_i}, \dots, y_k) = \begin{cases} P(y_1, \dots, \mathbf{y_i}, \dots, y_k) & \textit{if } \mathbf{y_i} = x; \\ \textbf{\textit{undefined}} & \textit{if } \mathbf{y_i} \neq x. \end{cases}$$

Definition 8 *An* atomic probabilistic selection condition *is an expression of the form* P *op* α*, where* $\alpha \in [0, 1]$ *and op* $\in \{=, \neq, \le, \ge, <, >\}$. Probabilistic selection conditions *are boolean combinations of* atomic probabilistic selection conditions.

Definition 9 *Let* $S = \langle T, V, P, C\rangle$ *be an SPO and let* $c : P$ *op* α *be a probabilistic atomic selection condition. Let* $\bar{x} \in dom(V)$*. The result of selection from* S *on* c *is defined as follows:* $\sigma_{P\ \mathit{op}\ \alpha}(S) = S' = \langle T, V, P', C\rangle$ *where*

S:

major: CS

DB	OS	Compilers	P
A	A	A	0.05
A	B	A	0.04
B	A	B	0.07
A	B	C	0.03
B	A	A	0.02
B	A	C	0.04
B	C	A	0.02
B	C	B	0.01
B	B	A	0.03
B	B	B	0.05
C	C	A	0.01
C	C	B	0.02
C	C	C	0.03

Logic = A

$\pi_{\text{DB,OS}}$**(S)** (step I):

major: CS

DB	OS	P
A	A	0.05
A	B	0.04
B	A	0.07
A	B	0.03
B	A	0.02
B	A	0.04
B	C	0.02
B	C	0.01
B	B	0.03
B	B	0.05
C	C	0.01
C	C	0.02
C	C	0.03

Logic = A

$\pi_{\text{DB,OS}}$**(S)** (step II):

major: CS

DB	OS	P
A	A	0.05
A	B	0.07
B	A	0.13
B	C	0.03
B	B	0.08
C	C	0.06

Logic = A

Figure 3. Projection on Probabilistic Table and on Probability values in SP-Algebra

$$P'(\bar{x}) = \begin{cases} P(\bar{x}) & \textit{if } P(\bar{x}) \textsf{ op } \alpha; \\ \textbf{\textit{undefined}} & \textit{otherwise.} \end{cases}$$

Example 3 *Figure 2 shows two examples of select queries on an SPO. The central object is obtained from the original SPO (left) as the result of the query* "Find all information about the probability of getting an A in Databases", *represented as* $\sigma_{\text{DB=A}}(S)$. *In the probability table of the resulting SPO, only the rows that have the value of the* DB *random variable equal to* A *will remain.*

The rightmost object in the figure is the result of the query "Find all grade combinations whose probability is greater than 0.11". *This query can be written as* $\sigma_{P>0.11}(S)$. *The probability table of the resulting object will contain only those rows from the original probability table where the probability value was greater than* 0.11.

Different selection operations commute, as shown in the following theorem:

Theorem 1 *Let* c *and* c' *be two (arbitrary) selection conditions and let* $\mathcal{S}$ *be a semistructured probabilistic relation. Then* $\sigma_c(\sigma_{c'}(\mathcal{S}) = \sigma_{c'}(\sigma_c(\mathcal{S}))$

4.2 Projection

Just as with selection, the results of *projection* operation differ, depending on which parts of an SPO are to be projected out.

Projection on *context* and *conditionals* is similar to the traditional relational algebra projection: either context attribute or a conditional is removed from an SPO object, which does not change otherwise. These operations change the semantics of the SPO and thus must be used with caution. However, it can be argued that removing attributes from the relations in a relational database system also changes the semantics of the data. Due to space restrictions, we do not describe such projections in further detail.

A somewhat more difficult and delicate operation is the projection on the set of *participating random variables*. A removal of a random variable from the SPO's *participant* set entails that information related to this random variable has to be removed from the probability table as well. Informally, this process corresponds to a process of removing one random variable from consideration in a joint probability distribution, which is usually called *marginalization*. The result of this operation is a new *marginal probability distribution*. It will be the purpose of our projection operation to compute this marginal probability distribution.

This computation can be performed in two steps. First, the columns for random variables that are to be projected out are removed from the probability table. In the remainder of the table, there can now be duplicate rows: rows that have all values (except for the probability value) coincide. All duplicate rows of the same type are then collapsed (coalesced) into one, with the new probability value computed as the sum of the values in the collapsed rows.

A formal definition of this procedure is given below.

Definition 10 *Let* $S = \langle T, V, P, C \rangle$ *be an SPO,* $V = \{v_1, \ldots, v_q\}$, $q > 1$ *and* $\mathcal{L}_V \subset V, \mathcal{L}_V \neq \emptyset$. *The projection of* S *on* $\mathcal{L}_V$*, denoted* $\pi_{\mathcal{L}_V}(S)$*, is defined to be an object* $S' = \langle T, \mathcal{L}_V, P', C \rangle$ *where* $P' : dom(\mathcal{L}_V) \longrightarrow [0,1]$ *and for each* $\bar{x} \in dom(\mathcal{L}_V)$,

$$P'(\bar{x}) = \sum_{\bar{y} \in dom(V - \mathcal{L}_V); P(\bar{x}, \bar{y}) \textit{is defined}} P(\bar{x}, \bar{y}).$$

Notice that projection on the set of participants is allowed only if the set of participants is **not a singleton** and

S:

college: Engineering

DB	OS	P
A	A	0.09
A	B	0.12
A	C	0.03
A	F	0.005
B	A	0.12
B	B	0.16
B	C	0.13
B	F	0.01
C	A	0.03
C	B	0.08
C	C	0.11
C	F	0.045
F	A	0
F	B	0.01
F	C	0.02
F	F	0.04

DS $\in$ { A, B }

$\mu_{\text{OS=A}}$(**S:**) (step I)

college: Engineering

DB	OS	P
A	A	0.09
B	A	0.12
C	A	0.03
F	A	0

DS $\in$ { A, B }

$\mu_{\text{OS=A}}$(**S:**) (step II)

college: Engineering

DB	P
A	0.375
B	0.5
C	0.125
F	0

DS $\in$ { A, B }
OS = A

Figure 4. Conditionalization in SP-Algebra.

if **at least** one random variable remains in the resulting set.

Example 4 *Figure 3 illustrates how projection on the set of participating random variables works. First, the columns of random variables to be projected out are removed from the probability table (step I). Next, the remaining rows are coalesced (step II). After the Compilers random variable had been projected out, the interim probability table has three rows (B,A) with probabilities 0.07, 0.02 and 0.04 respectively. These rows are combined into one row with probability value set to* $0.07 + 0.02 + 0.04 = 0.13$. *Similar operations are performed on the other rows.*

Theorem 2 *Let* $S = \langle T, V, P, C\rangle$ *be an SPO with* P *being a joint probability distribution of random variables* V. *Then, for* ***any*** $\emptyset \neq \mathcal{L}_V \subseteq V$, *the probability table* P' *from*

$$S' = \langle T, \mathcal{L}_V, P', C\rangle = \pi_{\mathcal{L}_V}(S)$$

contains the ***correct marginal probability distribution*** *of variables in* $\mathcal{L}_V$ *derived from* P.

4.3 Conditionalization

Conditionalization is an operation, specific to *probabilistic* algebras. Dey and Sarkar [9] were the first to consider this operation in the context of probabilistic databases.

Similarly to the projection operation, conditionalization reduces the probability distribution table. The difference is that the result of conditionalization is a *conditional probability distribution*. Given a joint probability distribution, conditionalization answers the following general query "What is the probability distribution of the remaining random variables if the value of some random variable v in the distribution is restricted to subset X of its values?"

Informally, conditionalization operation proceeds on a given SPO as follows. The input to the operation is one participating random variable of the SPO, v, and a subset of its values X. The first step of conditionalization consists of removing from the probability table of the SPO all rows whose v values are **not** from the set X. Then, the v column is removed from the table. Remaining rows are coalesced (if needed) in the same manner as in projection operation and the probability values are **normalized**. Finally, (v, X) is added to the set of conditionals of the resulting SPO.

The formal definition of conditionalization is given below. Note that if the original table is incomplete, there is no meaningful way to normalize a conditionalized probability distribution. Thus, we restrict this operation to situations where normalization is well-defined.

Definition 11 *An SPO* $S = \langle T, V, P, C\rangle$ *is* **conditionalization- compatible** *with an* atomic conditional selection condition $v = \{x_1, \ldots, x_h\}$ *iff*

- $v \in V$;
- *The restriction of* P *on* $\{x_1, \ldots, x_h\}$ *for* v *is a* **complete function**.

Definition 12 *Let* $S = \langle T, V, P, C\rangle$ *be an SPO which is* conditionalization-compatible *with an* atomic conditional selection condition $c : v = \{x_1, \ldots, x_h\}$.

The result of ***conditionalization*** *of* S *by c, denoted* $\mu_c(S)$, *is defined as follows:*

$$\mu_c(S) = \langle T, V', P', C' \rangle,$$

where

- $V' = V - \{v\}$;
- $C' = C \cup \{(v, \{x_1, \ldots, x_h\})\}$;
- $P' : V' \longrightarrow [0,1]$. *Let*

$$N = \sum_{\bar{y} \in dom(V')} \sum_{x \in \{x_1, \ldots, x_h\}} P(\bar{y}, x).$$

For any $\bar{y} \in dom(V')$,

$$P'(\bar{y}) = \frac{\sum_{x \in \{x_1, \ldots, x_h\}} P(\bar{y}, x)}{N}.$$

We can show that the definition above does indeed compute the conditional probability distribution.

Theorem 3 *Let* $S = \langle T, V, P, C \rangle$, $v \in V$ *and let* $c : v = \{x_1, \ldots, x_h\}$ *be an atomic selection condition. If* S *is* conditionalization-compatible *with c, then* $\mu_c(S)$ ***correctly computes the probability distribution*** *of the random variables in* $V - \{v\}$ *from* P*, under the assumption that* $v \in \{x_1, \ldots, x_h\}$.

Conditionalization can be extended to a semistructured relation in a straightforward manner. Given a relation $\mathcal{S}$, $\mu_c(\mathcal{S})$ will consist of $\mu_c(S)$ for each $S \in \mathcal{S}$ that is conditionalization-compatible with c.

Example 5 *Consider the SPO* S *defined in Example 1 describing the* joint probability distribution of performance in Databases and Operating Systems for College of Engineering majors who received either A or B in Data Structures. *Figure 4 depicts the work of the conditionalization operation* $\mu_{\mathsf{OS}=\mathsf{A}}(S)$. *Original object is shown to the left. As* $S.P$ *is a complete distribution,* S *is* conditionalization compatible *with* $\mathsf{OS} = \mathsf{A}$. *The first step of conditionalization consists of removing all rows that do not satisfy the conditionalization condition from* $S.P$ *(result depicted in the center). Then, on step II, the* OS *column is dropped from the table, probability values in the remaining rows are* normalized *and* $\mathsf{OS} = \mathsf{A}$ *is added to the list of conditionals. The rightmost object in Figure 4 shows the final result.*

4.4 Cartesian Product and Join

Cartesian product is defined only on pairs of *compatible* SPOs. Intuitively, a cartesian product of two probabilistic distributions is the joint probability distribution of random variables involved in both original distributions. Here, we will restrict ourselves to the assumption of independence between the probability distributions in cartesian products and joins. In our future research we will remove this restriction.

Intuitively, the SPOs are *compatible* for cartesian product if their participating variables are disjoint, but their conditionals **coincide**.

Definition 13 *Two SPOs* $S = \langle T, V, P, C \rangle$ *and* $S' = \langle T', V', P', C' \rangle$ *are* ***cartesian product-compatible (cp-compatible)*** *iff* $V \cap V' = \emptyset$ *and* $C = C'$.

We can now define the cartesian product.

Definition 14 *Let* $S = \langle T, V, P, C \rangle$ *and* $S' = \langle T', V', P', C' \rangle$ *are two cp-compatible SPOs. Then, the result of their cartesian product (under assumption of independence), denoted* $S \times S'$ *is defined as follows:*

$$S \times S' = S'' = \langle T'', V'', P'', C'' \rangle,$$

where

- $T'' = (T, T')$;
- $V'' = V \cup V'$;
- $P'' : dom(V'') \longrightarrow [0,1]$. *For all* $\bar{z} \in dom(V'')$; $\bar{z} = (\bar{x}, \bar{y})$; $\bar{x} \in dom(V), \bar{y} \in dom(V')$:

$$P''(\bar{z}) = P(\bar{x}) \cdot P'(\bar{y}).$$

- $C'' = C = C'$.

We can now define probabilistic joins. Two SPOs are *join-compatible* if they some share participating variables (these will be the "join attributes") and their conditionals coincide.

Definition 15 *Two SPOs* $S = \langle T, V, P, C \rangle$ *and* $S' = \langle T', V', P', C' \rangle$ *are* ***join-compatible*** *iff* $V \cap V' \neq \emptyset$ *and* $C = C'$.

In classical relational algebra the equijoin of two relations can be computed by first taking the cartesian product of the two relations, then selecting rows where the values of duplicate attributes coincide and then projecting out one of the duplicate attribute columns.

Our approach to a probabilistic join is very similar. Suppose $S = \langle T, V, P, C \rangle$ and $S' = \langle T', V', P', C'$ are two (join-compatible) SPOs to be joined. Let $V = V_1 \cup V_c$ and $V' = V_1' \cup V_c$, i.e. $V_c = V \cap V'$. Then, we want the join $S \bowtie S'$ of S and S' to contain the joint probability distribution of the set $V_1 \cup V_c \cup V_1'$. This distribution can be obtained in one of two ways:

1. Random variables V_c are projected out of S, and the marginal probability distribution obtained as a result is then used in a cartesian product with S'.

S:
major: CS

DB	OS	P
A	A	0.25
A	B	0.25
B	A	0.25
B	B	0.25

Logic = A

π_{DB}**(S):**
major: CS

DB	P
A	0.5
B	0.5

Logic = A

S$\bowtie_1$**S' = S**$\times\pi_{DS}$**(S')**
major: CS
college: ENGR

DB	OS	DS	P
A	A	A	0.125
A	A	B	0.125
A	B	A	0.125
A	B	B	0.125
B	A	A	0.125
B	A	B	0.125
B	B	A	0.125
B	B	B	0.125

Logic = A

S$\bowtie_2$**S' =**π_{DB}**(S)**$\times$ **S'**
major: CS
college: ENGR

DB	OS	DS	P
A	A	A	0.1
A	A	B	0.15
A	B	A	0.15
A	B	B	0.1
B	A	A	0.1
B	A	B	0.15
B	B	A	0.15
B	B	B	0.1

Logic = A

S':
college: ENGR

OS	DS	P
A	A	0.2
A	B	0.3
B	A	0.3
B	B	0.2

Logic = A

π_{DS}**(S'):**
college: ENGR

DS	P
A	0.5
B	0.5

Logic = A

Figure 5. Join in SP-Algebra

2. Random variables V_c are projected out of S' and the marginal probability distribution obtained as a result is then used in a cartesian product with S.

If the distribution of random variables from V_c in S and S' were the same, both methods would be equivalent. We, however, do not have this guarantee. The existence of different probability distributions on the same variables and the means of dealing with it presents an interesting and important problem which we are currently investigating. For now, we just accept that the results produced by the two procedures described above can be different. Thus we define **two** join operations: one favoring the variables in the first argument, and the other favoring the variables in the second.

Definition 16 *Let* $S = \langle T, V, P, C\rangle$ *and* $S' = \langle T', V', P', C'\rangle$ *be two join-compatible SPOs with* $V \cap V' = V_c$. *Then we define the results of two join operations on* S *and* S' *as follows:*

- $S \bowtie_1 S' = S \times \pi_{V'-V_c}(S')$.
- $S \bowtie_2 S' = \pi_{V-V_c}(S) \times S$.

Example 6 *Consider two simple SPOs* S *and* S' *as presented in Figure 5.* S *and* S' *share one random variable (OS) and their conditional parts coincide (Logic=A). Hence,* S *and* S' *are* join-compatible.

Computation of two joins of S *and* S', $S \bowtie_1 S'$ *and* $S \bowtie_2 S'$ *is presented in the rest of Figure 5. First, OS is projected out of* S *and* S' *(second column in Figure 5). Then, formulas* $S \bowtie_1 S' = S \times \pi_{V'-V_c}(S')$ *and* $S \bowtie_2 S' = \pi_{V-V_c}(S) \times S$ *are applied to compute respective joins (results are shown in the last two columns in Figure 5).*

In the resulting SPOs, the context will be a union of the contexts of the two original objects and the conditional part will be the same as in S *and* S'. *The probability table is formed by joining together the random variables from* S *and* $\pi_{DS}(S')$ *(* $\pi_{DB}(S)$ *and* S' *respectively) and multiplying the appropriate probabilities. For example, the probability value for the row A,B,A in* $S \bowtie_1 S'$ *is computed by multiplying the probability value from the row A,B of* S *(0.25) by the probability value from the row A of* $\pi_{DS}(S')$ *(0.5). In* $S \bowtie_2 S'$ *the probability value for the same row is computed by multiplying the probability value from the row A in* $\pi_{DB}(S)$ *(0.5) by the probability value from the row B,A in* S' *(0.3).*

5 Future Work

There are three major foci of our current work: (i) implementation of a semistructured DBMS based on this model; (ii) extension of the data model and the algebra to handle interval probabilities and (iii) the study of data fusion and conflict resolution problems that arise in this framework. In [13], the data model and SP-algebra are translated into XML [4]. This is the first step in the implementation of our DBMS.

6 Related Work

Cavallo and Pittarelli [6], Barbara, Garcia-Molina and Porter [2], Dey and Sarkar [9] and Lakshmanan et al. [15] have described different probabilistic relational models. Our work combines and extends the ideas contained in these works and applies them to a semistructured data model, which provides us with the added benefit of schema flexibility. Kornatzky and Shimony [14] and Eiter et al. [10] have developed probabilistic object models. Their approach is different from ours, as the probabilistic object (e.g. as described in [10]) represents a single real world entity with uncertain attribute values. In our case, an SPO represents a probability distribution of one of more random variables.

For more information about the Semistructured Data Model and its relationship to XML we refer the reader to [1, 5, 18]. Fernandez et al. [12] proposed recently a general purpose algebra for XML as a part of an effort to standardize querying of the XML data. This algebra formed the basis of W3C's XML Query Algebra Working Draft [11]. Both [12] and [11] focus on describing the semantics of *general purpose* queries to XML documents. In particular, they do not deal with querying probabilistic data. Also, the algebra presented here, works on the semistructured data irrespective of the format in which it is actually stored, while [12, 11] provide the syntax directed at XML data. The semistructured probabilistic algebra presented here has no data format-specific syntax. Future development will take this work as well as existing work on XML query languages [3] into account.

Aknowledgements

We'd like to thank the anonymous reviewers, whose suggestions improved this paper. We also want to thank V.S. Subrahmanian for helpful comments and the students working in the Bayesian Advisor group for their input at various stages, and their fabulous energy.

References

[1] S. Abiteboul, P. Buneman, D. Suciu. (1999) *Data on the Web: From Relations to Semistructured Data and XML*, Morgan Kaufmann.

[2] D. Barbara, H. Garcia-Molina and D. Porter. (1992) The Management of Probabilistic Data, *IEEE Trans. on Knowledge and Data Engineering*, Vol. 4, pps 487–502.

[3] A. Bonifati, S. Ceri (2000) Comparative Analysis of Five XML Query Languages. *SIGMOD Record*, vol 29, No. 1, pps 68–79.

[4] T. Bray, J. Paoli, C.M. Spreberg-McQueen (Eds.) (1998) Extensible Markup Language (XML) 1.0, *World Wide Web Consortium Recommendation*, http://www.w3.org/TR/1998/REC-xml-19980210.

[5] P. Buneman. (1997) Semistructured Data, in *Proc. PODS'97*, pp. 117–121.

[6] R. Cavallo and M. Pittarelli. (1987) The Theory of Probabilistic Databases, *Proc. VLDB'87*, pp. 71-81.

[7] A. Dekhtyar, J. Goldsmith, H. Li, B. Young. (2001) The Bayesian Advisor Project I: Modeling Academic Advising *University of Kentucky Dept. of Computer Science Technical Report # 323-01*.

[8] A. Dekhtyar and V.S. Subrahmanian. (2000) Hybrid Probabilistic Logic Programs, *Journal of Logic Programming*, vol 43, 3, pp. 187–250.

[9] D. Dey and S. Sarkar. (1996) A Probabilistic Relational Model and Algebra, *ACM Transactions on Database Systems*, Vol. 21, 3, pp. 339–369.

[10] T. Eiter, J. Lu, T. Lukasiewicz, V.S. Subrahmanian. (2001) Probabilistic Object Bases, accepted pending revisions to *ACM Transactions on Database Systems*.

[11] P. Fankhouser, M. Fernandez, A. Malhotra, M.Rys, J. Siméon, P. Wadler (Eds.) (2001) The XML Query Algebra. *World Wide Web Consortium Working Draft*, http://www.w3.org/TR/query-algebra/.

[12] M. Fernandez, J. Siméon, P. Wadler. (2000) An Algebra for XML Query, *in Proc. Found. of Softw. Theory and Theor. CS*, pps. 11-45.

[13] S. Hawkes, A. Dekhtyar (2001) Designing Markup Languages for Probabilistic Information, *University of Kentucky Tech. Report* TR 319-01.

[14] E. Kornatzky, S.E. Shimony. (1994) A Probabilistic Object Data Model, *Data and Knowledge Engineering*, vol 12, pp. 143–166.

[15] V.S. Lakshmanan, N. Leone, R. Ross and V.S. Subrahmanian. (1997) ProbView: A Flexible Probabilistic Database System. *ACM Transactions on Database Systems*, Vol. 22, Nr. 3, pps 419–469.

[16] R. Ng and V.S. Subrahmanian. (1993) Probabilistic Logic Programming, *Information and Computation*, 101, 2, pps 150–201.

[17] J. Pearl. (1988) *Probabilistic Reasoning in Intelligent Systems*, Morgan Kaufmann,

[18] D. Suciu. (1998) Semistructured Data and XML, in *Proc. 5th. Intl. Conf. on Foundation of Data Organization*, pps 1–12.