Journal of Intelligent Information Systems, , 1–39 (2004)

A Framework for Management of Semistructured Probabilistic Data

WENZHONG ZHAO wzhao0@cs.uky.edu
Department of Computer Science, University of Kentucky, Lexington, KY 40506-0046, USA

ALEX DEKHTYAR* dekhtyar@cs.uky.edu
Department of Computer Science, University of Kentucky, Lexington, KY 40506-0046, USA

JUDY GOLDSMITH goldsmit@cs.uky.edu
Department of Computer Science, University of Kentucky, Lexington, KY 40506-0046, USA

Editor:

Abstract. This paper describes the theoretical framework and implementation of a database management system for storing and manipulating diverse probability distributions of discrete random variables with finite domains, and associated information. A formal Semistructured Probabilistic Object (SPO) data model and a Semistructured Probabilistic Query Algebra (SP-algebra) are proposed. The SP-algebra supports standard database queries as well as some specific to probabilities, such as conditionalization and marginalization. Thus, the Semistructured Probabilistic Database may be used as a backend to any application that involves the management of large quantities of probabilistic information, such as building stochastic models. The implementation uses XML encoding of SPOs to facilitate communication with diverse applications. The database management system has been implemented on top of a relational DBMS. The translation of SP-algebra queries into relational queries are discussed here, and the results of initial experiments evaluating the system are reported.

Keywords: Probabilistic databases, Query algebras, Data models, Semistructured data

1. Introduction

Probabilistic information occurs in many vital applications, such as multimedia databases for storing the results of image recognition, logistics databases, stock market prediction software, and applications of Bayesian Nets [22]. We give an innovative approach to managing probabilistic information by treating the probability tables as the primary objects in the database. In order to make such data usable, we store significant auxiliary information along with the probability tables. Our databases are flexible enough to handle diverse "shapes" of tables over arbitrary numbers of discrete random variables. Our query algebra allows the user to retrieve data based on probabilities, variables, or associated information, and to transform the probability distributions according to the laws of probability theory.

Storing and managing probabilistic information has been an active research area in the last two decades. There have been relational [2, 5, 10, 19] and object [12, 18] data models proposed for storage and querying of probabilistic information. However, these approaches are not sufficiently flexible to handle different contexts in which probabilities must be dealt with in analyzing a stochastic system. For instance, consider auto insurance risk analysis, where the risk level of possible financial loss when offering a driver with an insurance

* To whom all correspondence should be addressed.

policy may be represented in variety of forms: a simple probability distribution for one aspect or a joint probability distribution for several aspects, or a simple or joint conditional probability distribution (risk level may depend on earlier driving record).

Information with different formats would require separate storage in any of the current probabilistic relational models, making even simple queries hard to express. For example, when one asks a query "Find all probability distributions that involve the aspect Driver's Age", the system has to query all the relations that have Driver's Age as a field. Note that this may require users to know in advance the names of tables that have this field and may result in thousands of separate queries. Thus we propose a new, *semistructured* probabilistic data model which alleviates this problem.

The semistructured data model [1, 4, 25] has gained wide acceptance recently as the means of representing data which do not conform to a rigid structure of schema. In particular, the similarity between the semistructured data model and the underlying data model for eXtensible Markup Language (XML) [3], the emerging open standard for data storage and transmission over the Internet, makes this approach attractive. This paper extends significantly the work presented in [7]. Here, we present the formal model for Semistructured Probabilistic Objects (SPOs) and a Semistructured Probabilistic Query Algebra (SP-algebra) for storing and managing probability distributions of discrete random variables with finite domains. We describe Semistructured Probabilistic DBMS, the XML-based implementation of this model. The results of our initial tests show that, even without query optimization, the SPDBMS performs well on SP-algebra queries.

In Section 2, we introduce a toy example, based on the auto insurance risk assessment process. Section 3 gives formal definitions of semistructured probabilistic objects, Section 4 introduces the underlying algebra for semistructured probabilistic databases, Section 5.1 shows how to represent this model in XML. Section 5 describes the pilot implementation and the test results of the performance of SPDBMS.

2. Motivating Example

In order to get car insurance, one must first fill out a complex form, giving information on driving history, insurance history, and a variety of personal matters. Based on this data, the insurer sets a policy premium for the available policies.

Insurers want to prevent major losses and maximize annual profits. As described by Russell and Norvig [24], a 1% improvement of risk assessment brings over a billion dollars annually for a typical insurance company. One way to lure customers is to lower prices; Most insurance companies try to set the insurance premium for each insurance policy holder as low as possible without giving up their profit.

How can an insurer increase the likelihood of a reasonable profit? Insurance companies could try to improve their risk assessment analysis, for instance, by constructing a Bayesian network which allows the company to decide what the financial risk level is for each policy holder.

Statistical information about the association between financial risk and driver personal information, driving skills and vehicle information can be obtained from a database of previous claims maintained by the company. Under the assumption that this information correctly reflects or approximates the true probabilities, it can assist in providing better estimates for policy premiums. However, the statistical information needs to be updated periodically so that it accurately reflects the current probabilities.

Consider a database to assist insurance companies with the risk assessment process. Note that the type of probabilistic information available to the insurance company in this example varies greatly. Figure 1 gives a Bayesian network model that includes many aspects that contribute to the risk level of a policy holder. The simplest is a *probability distribution of financial risk for one aspect*. The company may need the probability distribution of risk for Driver Age (DA) or the probability distribution over different rough values for risk given the number of years the driver has had a license (License Years (LY)).

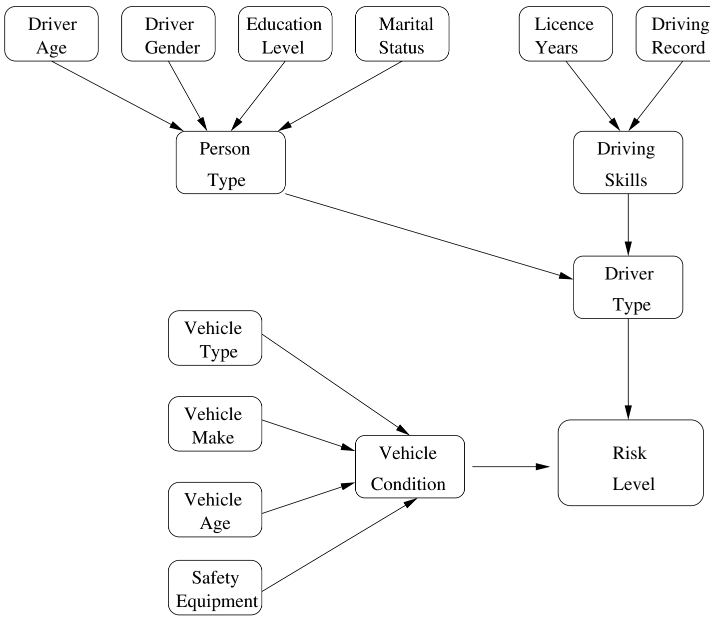

Figure 1. A Bayesian network of risk analysis for auto insurance companies.

Another type of probabilistic information that can be useful in this situation is a *joint probability distribution*. For instance, one might want to know the risk level of a customer who has college degree and a brand new passenger car. In this case, the company needs the probability distribution of risk for both Education Level (EL) and Vehicle Year (VY). This brings up another type of probabilistic information, the *conditional probability distribution*.

To make matters more complicated, we notice that the risk level can depend on her past Driving Record (DR) or other aspects observed. A Medium Accident in a Driving Record (DR) may suggest to the company that the policy holder might belong to the group of higher risk level. while a Yes in Safety Equipment (SE) might suggest that the policy holder might belong to the lower risk level group. Other information that can affect the probability distribution may include: where the policy holder lives, such as city, state, rural/urban; the policy holder's background such as race, employment type; vehicle information such as personal/business vehicle, etc.

The possible types of probabilistic information to be stored in a database for risk assessment support are shown in Figure 2. Note that here, and in all the work described in this paper, the domains of the variables are finite, as shown in Table 1.

Table 1. Representation of domain letter for random variables.

Domain Letter	A	B	C	F
Driver Age (DA)	<20	21-35	36-55	>56
Education Level (EL)	high school	college	advanced degree	none of above
Driver Gender (DG)	male	female		
Marital Status (MS)	single	married		
License Years (LY)	<1	1-3	3-10	>10
Driving Record (DR)	severe	medium	minor	none
Vehicle Type (VT)	heavy truck	light truck	passenger car	motorcycle
Vehicle Make (VM)	Toyota	BMW	Ford	GM
Vehicle Age (VA)	<1	1-5	5-10	>10
Safety Equipment (SE)	yes	no		

ω: S1

DA	P
A	0.4
B	0.1
C	0.2
F	0.3

ω: S2

DA	LY	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

ω: S3

city : Lexington
job : Managerial

DA	LY	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

ω: S4

city : Lexington
race : Asian

DA	LY	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

SE = B
DR $\in$ { A, B }

Figure 2. Different types of probabilistic information to be stored in the database for risk analysis applications (from left to right: single variable probability distribution, joint probability distribution of 2 variables, joint probability distribution with context, and conditional joint probability distribution with context.)

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 [15]. Probabilistic relational models [2, 10, 19] lack the flexibility to store all of our data in a straightforward manner. In the risk analysis application the aspects are viewed as *random variables*. As such, it is natural to represent each aspect as a database attribute that can take values from its domain. However, with such an interpretation, a joint probability distribution of values in two aspects will have a schema different from the joint probability distribution of three aspects, and therefore, will have to be stored in a separate relation. In such a database, expressing queries like "Find all probability distributions that include **Driver Age** as a random variable" is very inconvenient, if at all possible.

Probabilistic Object models [12, 18] are also not a good fit for storing this kind of data. In the framework of Eiter et al. [12], 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.

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 $\mathcal{R} = \{\mathsf{city}, \mathsf{job}, \mathsf{race}\}$, the following are valid semistructured schemas over $\mathcal{R}$: $R^\star_1 = \{\mathsf{city}, \mathsf{job}\}$; $R^\star_2 = \{\mathsf{city}, \mathsf{city}, \mathsf{job}, \mathsf{race}\}$; $R^\star_3 = \{\mathsf{job}, \mathsf{job}, \mathsf{job}\}$.

Let $\mathcal{P}$ denote a *probability space* used in the framework to represent probabilities of different events. Examples of such probability spaces include (but are not limited to) the interval $[0, 1]$ and the set $\mathsf{C[0,1]}$ of *all subintervals of* $[0, 1]$ [20, 8, 19]. For each probability space $\mathcal{P}$ there should exist a notion of a *consistent probability distribution over* $\mathcal{P}$[1].

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 tuple $S = \langle T, V, P, C, \omega\rangle$, 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 ***conditional*** of S.
- ω, called the ***path expression***, is an expression of Semistructured Probabilistic Algebra (SP-Algebra).

An explanation of this definition is in order. For our data model to possess the ability to store all the probability distributions mentioned in Section 2 (see Figure 2), 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*. A 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. **Conditional.** A probability table may represent a *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.

5. **Origin** or **path** of the object. Each SPO in an SP-Database can either be inserted into the database directly, or can be a result of one or more SP-Algebra operations over already existing SPOs. When an SPO is *inserted* into the database, a unique identifier is assigned as its path. Whenever an SPO is created as a result of an SP-Algebra operation, its path is extended by the description of this operation. An SPO inserted into the database is called a *base SPO*. In Section 4 we introduce the syntax for complex path expressions that are formed when SP-Algebra operations are performed on SPOs.

Intuitively, a Semistructured Probabilistic Object represents a (possibly complex) probability distribution and the information associated with it. The actual distribution is described by the participating random variables and probability table parts of the object. The conditional part, when non-empty, indicates that the object represents a conditional probability distribution and specifies the conditions. The context contains any *non-stochastic* information associated with the distribution. Finally its path tells us how this object has been constructed. If the path is atomic (single unique identifier), than the object had been *constructed from scratch* and inserted into the database. Complex paths indicate which database objects participated in its creation and what SP-Algebra operations have been applied to obtain it. As examples throughout the paper will show, knowing how an object was constructed may help in the interpretation of its probability table.

EXAMPLE: Consider the joint probability distribution of risk based on Driver Age (DA) and License Years (LY) for Asian drivers in Lexington who had either a severe or medium accident within the last 3 years, as defined in Figure 3.

We can break this information into our five constituent parts as follows:

ω: S1

race : Asian
city : Lexington

DA	LY	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	0.0
F	B	0.01
F	C	0.02
F	F	0.04

DR $\in$ { A, B }

Figure 3. Joint Probability Distribution of risk on Driver Age and License Years for Asian drivers in Lexington.

participating random variables: $V = \{\mathsf{DA}, \mathsf{LY}\}$.

probability table: as shown in Figure 3, the probability table defines a *complete* and *consistent* probability distribution.

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

context: information about where the driver lives and the driver's race is **not** represented by random variables in our universe. They are, therefore, represented as relational attributes city and race, respectively. Thus, city: Lexington and race: Asian are the **context** of the probabilistic information in this example.

path: assuming that this information is being added to the database, we associate with this SPO a unique identifier S1 that will serve as its path.

□

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 (SP-relation)***. A finite collection $\mathcal{D}_S = \{\mathcal{S}_1, \ldots, \mathcal{S}_m\}$ is called a ***semistructured probabilistic database (SP-database)***.

One important difference between *semistructured probabilistic databases* and classic relational or relational probabilistic databases is that each table in a relational database has a specified schema whereas all SP-relations are "schema-less": any collection of SPOs can form an SP-relation. Thus, the division of a semistructured probabilistic database into relations is a matter of the logic of a particular application. For example, if the SP-database is built from the information supplied by three different experts, this information can be arranged into three semistructured probabilistic relations according to the origin of each object inserted in the database. Alternatively, the information can be arranged in SP-relations by the date it was obtained.

The key to the efficient use of semistructured probabilistic databases in representing probabilistic information is the management of data stored in SPDs. In *probabilistic relational databases*, Barbara et al. [2], Dey and Sarkar [10] and Lakshmanan et al. [19] define probabilistic relational algebras by extending the classical relational algebra. They add probability-specific (and probability theory compliant) manipulation of the probabilistic attributes in the relations. We also define a new semistructured probabilistic algebra 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 contains three standard set operations, union, intersection and difference and extends the definitions of standard relational operations selection, projection, Cartesian product and join to account for the appropriate management and maintenance of probabilistic information within SPOs. In addition, a new operation, conditionalization (see also [10]), is defined in SP-algebra. This operation is specific to the probabilistic databases and results in the construction of SPOs that represent conditional probability distributions of the input SPOs.

Before proceeding with the description of individual operations, we need to make an important distinction between the notions of *equality* and *equivalence* of SPOs. Two SPOs S and S' are equal if all their components, *including the paths* are equal. At the same time, only the first four components of any SPO: context, participating variables, probability table and conditional information represent the real content of the object. The path merely records how the object was obtained in the database. It is possible to obtain, as a result of SP-Algebra operations, two SPOs with the same first four components but different paths. Such objects, will not, technically, be equal. However, they would represent exactly the same information, and in many cases, we could substitute one such object with another without any loss. We reserve the notion of *equivalence* of SPOs for such situations.

Definition 2. Let $S = \langle T, V, P, C, \omega \rangle$ and $S' = \langle T', V', P', C', \omega' \rangle$ be two SPOs. S *is equivalent to* S', denoted $S \equiv S'$ iff $T = T'$, $V = V'$, $P = P'$ and $C = C'$.

4.1. Set Operations

Semistructured Probabilistic relations are sets of SPOs. Because of it, the definitions of *union*, *intersection* and *difference* of SP-relations are straightforward.

Definition 3. Let $\mathcal{S}$ and $\mathcal{S}'$ be two SP-relations. Then,

- Union: $\mathcal{S} \bigcup \mathcal{S}' = \{S | S \in \mathcal{S} \text{ or } S \in \mathcal{S}'\}$.
- Intersection: $\mathcal{S} \bigcap \mathcal{S}' = \{S | S \in \mathcal{S} \text{ and } S \in \mathcal{S}'\}$.
- Difference: $\mathcal{S} - \mathcal{S}' = \{S | S \in \mathcal{S} \text{ and } S \notin \mathcal{S}'\}$.

We note two features of the set operations in SP-Algebra. Classical relational algebra has a restriction on the applicability of the set operations: they are defined only on pairs of relations with matching schemas. Because SP-relations are schema-less and represent logical rather than syntactic groupings of probability distributions in an SP-database, set operations are applicable to any pair of SP-relations. Another feature is that set operations do not leave their imprint on the *path* component of individual SPOs.

4.2. Selection

Given an SPO $S = \langle T, V, P, C, \omega \rangle$, a selection query may be issued to any part except the path. Each part requires its own language of selection conditions.

Given an SPO S, selection on *context*, *participating random variables* and *conditionals* will result in either S being selected or not in its entirety (as is the case with selection on classical relations). It is also possible to select a subset of rows of the *probability table* based either on the values of random variables or on the probability values of individual rows in the probability table. Such selection operations may return only *part* of the original probability table P, while keeping context, conditionals and participating random variables intact. For any selection operation, the path expression of the result will be updated to include the selection operation.

The five different types of selections are illustrated in the following example.

EXAMPLE: Consider the SPO S described in Example 1. Five different types of selection queries are illustrated below.

1. "Find all probability distributions related to Asian drivers." $S.T$ contains the tuple race : Asian, therefore S matches the selection condition.
2. "Find all probability distributions that involve the Driver Age aspect." As DA is one of the participating random variables of S, S matches the selection condition.
3. "Find all probability distributions related to drivers who had a medium or severe accident within the last 3 years". The conditional part of S contains expression $\mathsf{DR} \in \{\mathsf{A}, \mathsf{B}\}$ which matches the selection condition ("medium or severe accident within the last 3 years").
4. "What information is available about the risk when offering insurance to a driver with less than one year of driving experience?" $S.P$ contains four entries that relate to the probability for drivers with less than one year of driving experience. 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.

   As an alternative, consider the query, "what is the risk when offering insurance to a 30-year-old driver with 5 years of driving experience?" The answer to this query on S would contain only one line from $S.P$, for the appropriate information of drivers).
5. "What outcomes have probability over 0.1?" In the probability table of S, there are five possible outcomes that have probability greater than 0.1. In the result of executing this query on S, $S.P$ should contain exactly these five rows, with $S.T$, $S.V$ and $S.C$ remaining unchanged.

□

4.2.1. Selection on Context, Participating variables 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 4.

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

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

Definition 5. Let $S = \langle T, V, P, C, \omega \rangle$ be an SPO and let $c : Q(A, x)$ be an atomic context selection condition. Let $\omega' = \sigma_c(\omega)$ and let $S' = \langle T, V, P, C, \omega' \rangle$. Then $\sigma_c(S) = \{S'\}$ if and only if

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

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

Definition 6. Let $S = \langle T, V, P, C, \omega \rangle$ be an SPO and let $c : v \in V$ be an atomic participation selection condition. Let $\omega' = \sigma_c(\omega)$ and let $S' = \langle T, V, P, C, \omega' \rangle$. Then $\sigma_c(S) = \{S'\}$ if and only if $v \in V$.

Definition 7.

1. Let $S = \langle T, V, P, C, \omega \rangle$ be an SPO and let $c : u = \{x_1, \ldots, x_h\}$ be an atomic conditional selection condition. Let $\omega' = \sigma_c(\omega)$ and let $S' = \langle T, V, P, C, \omega' \rangle$. Then $\sigma_c(S) = \{S'\}$ if and only if $C \ni (u, X)$ and $X = \{x_1, \ldots, x_h\}$.
2. Let $c : u \ni x$ be an atomic conditional selection condition. Then $\sigma_c(S) = \{S'\}$ if and only if $C \ni (u, X)$ and $X \ni x$.

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

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 $S = \langle T, V, P, C, \omega \rangle$, attribute A must be present in $S.T$. If A is not in $S.T$, 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.

4.2.2. Selection on Probability Table or Probabilities Selection operations considered in the previous sections 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. In both 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 8. 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 9. Let $S = \langle T, V, P, C, \omega \rangle$ be an SPO, $V = \{v_1, \ldots, v_k\}$ and let $c : v = x$ be an atomic probabilistic table selection condition.

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

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

and $\omega' = \sigma_c(\omega)$.

Definition 10. An *atomic probabilistic selection condition* is an expression of the form $P \ \mathsf{op} \ \alpha$, where $\alpha \in [0, 1]$ and $\mathsf{op} \in \{=, \neq, \leq, \geq, <, >\}$. *Probabilistic selection conditions* are Boolean combinations of *atomic probabilistic selection conditions*.

Definition 11. Let $S = \langle T, V, P, C, \omega \rangle$ be an SPO and let $c : P \ \mathsf{op} \ \alpha$ be an atomic probabilistic selection condition. Let $\bar{x} \in dom(V)$. The result of selection from S on c is defined as follows: $\sigma_{P \ \mathsf{op} \ \alpha}(S) = S' = \langle T, V, P', C, \omega' \rangle$ where

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

and $\omega' = \sigma_c(\omega)$.

EXAMPLE: Figure 4 shows two examples of selection 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 risk when offering insurance to a 19-year-old driver"*, denoted $\sigma_{\mathsf{DA}=\mathsf{A}}(S)$. In the probability table of the resulting SPO, only the rows that have the value of the DA random variable equal to A remain.

ω: **S**		
race:	Asian	
DA	LY	P
A	A	0.10
A	B	0.10
B	A	0.13
B	C	0.09
C	C	0.16
DR = B		

ω: $\sigma_{\mathsf{DA}=\mathsf{A}}$**(S)**		
race: Asian		
DA	LY	P
A	A	0.10
A	B	0.10
DR = B		

ω: $\sigma_{\mathsf{P}>0.11}$**(S)**		
race: Asian		
DA	LY	P
B	A	0.13
C	C	0.16
DR = B		

Figure 4. Selection on Probabilistic Table and on Probability values in SP-Algebra

The rightmost object in the figure is the result of the query *"Find all 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.

□

SP-Algebra operations can be extended to a semistructured probabilistic relation, as described in the following proposition.

PROPOSITION 1 *Any SP-Algebra operation on a semistructured probabilistic relation is equivalent to the union of the SP-Algebra operation on each SPO in the SP-relation.*

- *Let $\mathcal{S}$ be a semistructured probabilistic relation and γ be one of the three unary SP-Algebra operators. Then $\gamma(\mathcal{S}) = \bigcup_{S \in \mathcal{S}}(\gamma(S))$.*

- *Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two semistructured probabilistic relations and $\otimes$ be one of the two binary SP-Algebra operators. Then $\mathcal{S}_1 \otimes \mathcal{S}_2 = \bigcup_{S_1 \in \mathcal{S}_1} \bigcup_{S_2 \in \mathcal{S}_2}(S_1 \otimes S_2)$.*

Different selection operations commute, as shown in the following theorem. Proofs for all theorems are provided in Appendix A.

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

4.3. 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 a 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.

Definition 12. Let $S = \langle T, V, P, C, \omega \rangle$ be an SPO and let $\mathcal{L} \subset \mathcal{A}$ be a set of context attributes. The projection of S onto $\mathcal{L}$, denoted $\pi_{\mathcal{L}}(S)$ is an SPO $S' = \langle T', V, P, C, \omega' \rangle$ where

- $T' = \{(A, x) | (A, x) \in T, A \in \mathcal{L}\}$, i.e., T' contains all entries from T for attributes from the list $\mathcal{L}$ only.
- $\omega' = \pi_{\mathcal{L}}(\omega)$.

Definition 13. Let $S = \langle T, V, P, C, \omega \rangle$ be an SPO and let $\mathcal{F}$ be a set of conditionals. The projection of the conditional part of S onto $\mathcal{F}$, denoted $\pi_{c:\mathcal{F}}(S)^2$ is an SPO $S' = \langle T, V, P, C', \omega' \rangle$ where

- $C' = \{(u, X) | (u, X) \in C, u \in \mathcal{F}\}$.
- $\omega' = \pi_{c:\mathcal{F}}(\omega)$.

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 corresponds to 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* that needs to be stored in the probability table component of the resulting SPO.

This computation is 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 exist duplicate rows whose values for all the fields except the probability 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 14. Let $S = \langle T, V, P, C, \omega \rangle$ 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 $S' = \langle T, \mathcal{L}_V, P', C, \omega' \rangle$ where $P' : dom(\mathcal{L}_V) \longrightarrow [0, 1]$ and for each $\bar{x} \in dom(\mathcal{L}_V)$,

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

and $\omega' = \pi_{\mathcal{L}_V}(\omega)$.

Notice that projection on the set of participants is allowed only if the set of participants is not a singleton and if at least one random variable remains in the resulting set.

EXAMPLE: Figure 5 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 Vehicle Type (VT) random variable has 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. □

ω: S

race: Asian

DA	LY	VT	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

DR = B

ω: $\pi_{\text{DA,LY}}$(S) (step I)

race: Asian

DA	LY	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

DR = B

ω: $\pi_{\text{DA,LY}}$(S) (step II)

race: Asian

DA	LY	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

DR = B

Figure 5. Projection on Probabilistic Table and on Probability values in SP-Algebra

4.4. Conditionalization

Conditionalization is an operation specific to *probabilistic* algebras. Dey and Sarkar [10] 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, the 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. The remaining rows are coalesced (if needed) in the same manner as in the 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 15. An SPO $S = \langle T, V, P, C, \omega \rangle$ is ***conditionalization-compatible*** with an *atomic conditional selection condition* $v = \{x_1, \ldots, x_h\}$ iff

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

Definition 16. Let $S = \langle T, V, P, C, \omega \rangle$ 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:

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

where

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

  Let
  $$N = \sum_{\bar{y} \in dom(V')} \sum_{\bar{x} \in \{x_1, \ldots, x_h\}} P(\bar{x}, \bar{y}).$$
  For any $\bar{y} \in dom(V')$,
  $$P'(\bar{y}) = \frac{\sum_{\bar{x} \in \{x_1, \ldots, x_h\}} P(\bar{x}, \bar{y})}{N};$$
- $\omega' = \mu_c(\omega)$.

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. SPOs not conditionalization-compatible with c will not be included in $\mu_c(\mathcal{S})$.

EXAMPLE: Consider the SPO S defined in Example 1 describing the *joint probability distribution of risk on Driver Age (DA) and License Years (LY) for Asian drivers in Lexington who had either a severe or medium accident within the last 3 years*. We try to derive the probability distribution for drivers with less than one year of driving experience. Figure 6 depicts the work of the conditionalization operation $\mu_{\mathsf{LY}=\mathsf{A}}(S)$. The original object is shown to the left. As $S.P$ is a complete distribution, S is *conditionalization compatible* with LY $=$ 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 LY column is dropped from the table, probability values in the remaining rows are *normalized* and LY $=$ A is added to the list of conditionals. The rightmost object in Figure 6 shows the final result. □

ω: **S**

race: Asian

DA	LY	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

DR $\in$ { A, B }

ω: $\mu_{\mathsf{LY}=\mathsf{A}}$(**S**) (step I)

race: Asian

DA	LY	P
A	A	0.09
B	A	0.12
C	A	0.03
F	A	0

DR $\in$ { A, B }

ω: $\mu_{\mathsf{LY}=\mathsf{A}}$(**S**) (step II)

race: Asian

DA	P
A	0.375
B	0.5
C	0.125
F	0

DR $\in$ { A, B }
LY = A

Figure 6. Conditionalization in SP-Algebra.

4.5. *Cartesian Product*

Sometimes an SP database has only simple probability distributions for some random variables. In order to get a joint probability distribution, either a Cartesian product or join operation has to be performed on the SPOs storing these distributions. Intuitively, a Cartesian product or join of two probabilistic distributions is the joint probability distribution of random variables involved in both original distributions. The Cartesian product is defined only on pairs of *compatible* SPOs. Here, we will restrict ourselves to the assumption of independence between the probability distributions in Cartesian products. This restriction allows us to represent the result as a point probability distribution[3].

Two SPOs are *compatible* for Cartesian product if their participating variables are disjoint, but their conditionals coincide. When the sets of participating variables are not disjoint, we will use the join operation instead of Cartesian product to find, for instance, the Driver's joint probability distribution.

Definition 17. Two SPOs $S = \langle T, V, P, C, \omega \rangle$ and $S' = \langle T', V', P', C', \omega' \rangle$ are **Cartesian product-compatible** (**cp-compatible**) if and only if $V \cap V' = \emptyset$ and $C = C'$.

We can now define the Cartesian product.

Definition 18. Let $S = \langle T, V, P, C, \omega \rangle$ and $S' = \langle T', V', P', C', \omega' \rangle$ are two cp-compatible SPOs. Then, the result of their Cartesian product (under assumption of independence), denoted $S \times S'$, is defined as follows:

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

where

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

4.6. *Join*

Join is also defined only on pairs of compatible SPOs. Two SPOs are *join-compatible* if they share some participating variables (these will be the "join attributes") and their conditionals coincide.

Definition 19. Two SPOs $S = \langle T, V, P, C, \omega \rangle$ and $S' = \langle T', V', P', C', \omega' \rangle$ are **join-compatible** if and only if $V \cap V' \neq \emptyset$ and $C = C'$.

Given two join-compatible SPOs S and S', we can break the set $V \cup V'$ into three non-empty disjoint parts: $V_1 = V - V'$, $V_1' = V' - V$ and $V_c = V \cap V'$. The information about the probability distribution of random variables in V_c can be found in both S and S'. The join operation must take this into consideration when the joint probability distribution for variables in $V \cup V'$ is computed. The key to computing the joint distribution correctly is the following statement.

LEMMA 1 *Let $\bar{x} \in dom(V_1)$, $\bar{y} \in dom(V_c)$, $\bar{z} \in dom(V_1')$, and let V_1, V_c and V_1' all be disjoint. Under the assumption of independence between variables in V_1 and V_1',*

$$P(\bar{x}, \bar{y}, \bar{z}) = P(\bar{x}, \bar{y}) \cdot P(\bar{y}, \bar{z})/P(\bar{y}) = P(\bar{x}, \bar{y}) \cdot P(\bar{z}|\bar{y}) = P(\bar{z}, \bar{y}) \cdot P(\bar{x}|\bar{y}).$$

We can now define the join operations. We want the join of S and S' to contain the joint probability distribution of the set $V_1 \cup V_c \cup V_1'$. Since $P(\bar{y})$ could be obtained either from S or from S', there exist two families of join operations, called left join, $\ltimes$, and right join, $\rtimes$, with the following definitions. The only difference between the two join operations is the probability distribution.

Definition 20. Let $S = \langle T, V, P, C, \omega \rangle$ and $S' = \langle T', V', P', C', \omega' \rangle$ be two *join-compatible* SPOs. Let $V = V_1 \cup V_c$ and $V' = V_1' \cup V_c$, i.e. $V_c = V \cap V'$. We define the operations of left join of S and S', denoted $S \ltimes S'$ and right join of S and S', denoted $S \rtimes S'$ as follows:

$$S \ltimes S' = S'' = \langle T'', V'', P'', C'', \omega'' \rangle;$$
$$S \rtimes S' = S''' = \langle T'', V'', P''', C'', \omega''' \rangle,$$

where

- $T'' = T \cup T'$;
- $V'' = V_1 \cup V_c \cup V_1'$;
- $P'', P''' : dom(V'') \longrightarrow [0, 1]$.
  For all $\bar{w} \in dom(V'')$; $\bar{w} = (\bar{x}, \bar{y}, \bar{z})$; $\bar{x} \in dom(V_1)$, $\bar{y} \in dom(V_c)$, $\bar{z} \in dom(V_1')$:
  $$P''(\bar{w}) = P(\bar{x}, \bar{y}) \cdot P'(\bar{y}, \bar{z})/P'(\bar{y});$$
  $$P'''(\bar{w}) = P(\bar{x}, \bar{y}) \cdot P'(\bar{y}, \bar{z})/P(\bar{y}).$$
- $C'' = C = C'$.
- $\omega'' = \omega \ltimes \omega'$; $\omega''' = \omega \rtimes \omega'$.

Two *join-compatible* SPOs are *join-consistent* if probability distributions on the set of shared participating variables are identical for both SPOs.

Definition 21. Let $S = \langle T, V, P, C, \omega \rangle$ and $S' = \langle T', V', P', C', \omega' \rangle$ be two *join-compatible* SPOs with $V \cap V' = V_c$. Then, S and S' are **join-consistent** iff $P(\bar{y}) = P'(\bar{y})$ for any $\bar{y} \in dom(V_c)$.

EXAMPLE: Consider two simple SPOs S and S' as presented in Figure 7. S and S' share one random variable (LY) and their conditional parts coincide (DR = B). Hence, S and S' are *join-compatible*.

The results of the two join operations of S and S', $S \ltimes S'$ and $S \rtimes S'$, are presented in the rest of Figure 7. In the resulting SPOs, the context will be a union of the contexts of the two

ω: **S**

race: Asian

DA	LY	P
A	A	0.25
A	B	0.25
B	A	0.25
B	B	0.25

DR = B

ω: π_{LY}**(S)**

race: Asian

LY	P
A	0.5
B	0.5

DR = B

ω: **S**$\ltimes$**S'**

race: Asian
city: Lexington

DA	LY	VT	P
A	A	A	0.1
A	A	B	0.1
A	B	A	0.15
A	B	B	0.15
B	A	A	0.1
B	A	B	0.1
B	B	A	0.15
B	B	B	0.15

DR = B

ω: **S**$\rtimes$**S'**

race: Asian
city: Lexington

DA	LY	VT	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

DR = B

ω: **S'**

city: Lexington

LY	VT	P
A	A	0.2
A	B	0.2
B	A	0.3
B	B	0.3

DR = B

ω: π_{LY}**(S')**

city: Lexington

LY	P
A	0.4
B	0.6

DR = B

Figure 7. Join operations in SP-Algebra

original objects and the conditional part will be the same as in S and S'. The probability table is formed by first projecting the shared variable set from one of the original SPOs. For left join, do projection on all the variables in V_c , in this case LY, against the right operand SPO S', and save the result in a temporary SPO object *temp*,

$$temp = \pi_{V_c}(S'),$$

as shown in the center of the figure. Then, formulate the probability table by using the definition for left join, which is $P''(\bar{x}, \bar{y}, \bar{z}) = P(\bar{x}, \bar{y}) \times P'(\bar{y}, \bar{z})/temp.P'(\bar{y})$. The right join can be computed in the same manner.

Respective join results are shown in the last two columns in Figure 7. One can see that these two SPOs are not join-consistent.

□

4.7. *Semantics of SP-Algebra Operations*

The problem of determining the meaning of the results of the operations of SP-Algebra is complicated by the fact that at any moment, SP-databases can contain SPOs of two types. In the SPOs of the first type, the probabilities of all rows are exact, while in the SPOs of the second type, the probabilities of some rows may represent the lower bounds on the probability of those instances. We start this section by defining the two types of SPOs formally, discussing their properties and the effects that different SP-Algebra operations have on the SPOs in light of this.

Definition 22. An SPO $S = \langle T, V, P, C, \omega \rangle$ is a *Type I* SPO iff

$$\sum_{\bar{x} \in dom(V)} P(\bar{x}) = 1.$$

Otherwise, S is a *Type II* SPO.

When S is a Type I SPO, its probability table is *complete*: the probabilities of all rows add up to *exactly* 1. The probability table may contain a row for every instance $\bar{x} \in dom(V)$, or it may omit some of the instances. However, because the probabilities of the rows present in the table add up to 1, we know that the probabilities of all omitted rows are 0, and these can be added to the probability table of S. Basically, when S is a Type I SPO, we are guaranteed that *for all* $\bar{x} \in dom(V)$, $P(\bar{x})$ *is the exact point probability of instance* $\bar{x}$.

The nature of Type II SPOs is somewhat more complex. The fact that the sum of probabilities in all rows of the probability table is less than 1 means that the probability table is missing some information. This can either be *missing instances*: some $\bar{x} \in dom(V)$ has a non-zero probability but is not included in the probability table of S, or *underestimation*: all possible instances are present, but the probabilities add up to less than 1, which means that information about the probabilities of some (possibly all) instances presents only a *lower bound* on the true probability of the instance in the distribution.

It is important to note here that SP-Algebra operations allow for Type II SPOs to occur in the SP-database, even if all original SPOs in the database were Type I. We illustrate this on the following example.

EXAMPLE: Consider the SPO S, the left-most SPO depicted in Figure 8. Note that several rows are missing from the tables, namely those with probability 0. This is done for eesthetic reasons; the full tables are represented in the XML format and in the underlying relational tables. It is clear that $dom(\mathsf{DA}) = dom(\mathsf{LY}) = \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$, which means that not all instances are present in the probability table P of S. However, because the probabilities of all rows present in P add up to exactly 1, S is a *Type I* SPO and the probabilities of all instances not in P are 0. We also can be assured that each probability is exact.

Consider now the central SPO $S' = \sigma_{P \leq 0.2}(S)$ in Figure 8. Here, only the rows with probability value less than or equal to 0.2 are selected from the probability table of S. There are 3 such rows, for a combined probability of 0.35. Therefore, S' is a *Type II* SPO. We note here that, despite being of Type II, the probability of each row is exact. Consider now the SPO $S'' = \pi_{\mathsf{LY}}(S') = \pi_{\mathsf{LY}}(\sigma_{P \leq 0.2}(S))$ shown on the right side of Figure 8. The projection operation leads to removal of the DA random variable from S''. However, because each row of P' had a different value for LY, P'' will have three rows, one for each value of LY: A,B and C. While the probability table P'' of S'' has no missing rows, the probabilities add up to the same value of 0.35 as in P', and therefore S'' is also a *Type II* SPO. More importantly, the rows for A and C contain *incomplete* probability — applying π_{LY} to S we can see that the probability of getting the grade of A in LY is 0.43 and the probability of getting the grade of C is 0.37. Therefore, the probability values in the probability table P'' represent only *the lower bounds* on the probabilities of these rows.

□

ω: **S**

race: Asian

DA	LY	P
A	A	0.1
A	B	0.2
B	A	0.33
B	C	0.22
C	C	0.15

DR = B

ω: $\sigma_{\mathsf{P} \leq 0.2}$**(S)**

race: Asian

DA	LY	P
A	A	0.1
A	B	0.2
C	C	0.15

DR = B

ω:$\pi_{\mathsf{LY}}(\sigma_{\mathsf{P} \leq 0.2}$**(S))**

race: Asian

LY	P
A	0.1
B	0.2
C	0.15

DR = B

Figure 8. Selection on Probabilistic Table and on Probability values in SP-Algebra

The difference in the meaning of probability values for Type I and Type II SPOs causes us to apply extra caution when interpreting the results of SP-Algebra operations. In particular, when considering a specific SP-Algebra operation applied to an SPO or a pair of SPOs, it is important for us to know the type of the input objects and be able to determine the type of the result. The following proposition identifies the set of "safe" operations in SP-Algebra: operations that given Type I SPOs are guaranteed to produce Type I results.

PROPOSITION 2 *Let S and S' be two Type I SPOs. Then, the following SPOs are also Type I:*

1. $\sigma_c(S)$, *where c is a selection condition on* context, participating random variables *or* conditional.

2. $\pi_{\mathcal{L}}(S)$, $\pi_{c:\mathcal{F}}(S)$ *and* $\pi_{\mathcal{L}_V}(S)$, *where $\mathcal{F}$ is a list of context attribute names and $\mathcal{F}$, $\mathcal{L}_V \subset \mathcal{V}$.*

3. $\mu_c(S)$, *where c is a conditional selection condition.*

4. $S \times S'$.

5. $S \ltimes S'$ *and* $S \rtimes S'$.

Two operations missing from the list in Proposition 2 are selection on probabilities and selection on probability table. Figure 8 shows how selection on probabilities can produce a Type II SPO from Type I input; selection on probability table can be used to obtain the same result.

The following statements specify the semantics of the SP-Algebra operations producing Type I results.

THEOREM 2 *Let $S = \langle T, V, P, C, \omega \rangle$ be a Type I SPO and let $\emptyset \neq \mathcal{L}_V \subset V$. Let $S' = \langle T, \mathcal{L}_V, P', C, \omega' \rangle = \pi_{\mathcal{L}_V}$. Then P' contains the correct marginal probability distribution of random variables in $\mathcal{L}_V$ given the probability distribution P.*

THEOREM 3 *Let $S = \langle T, V, P, C, \omega\rangle$ be a Type I SPO and let c be a conditional selection condition involving variable $v \in V$. Let $S' = \langle T, V - \{v\}, P', C', \omega'\rangle = \mu_c$. Then P' contains the correct conditional probability distribution of random variables $V - \{v\}$ from the distribution P given condition c on v.*

THEOREM 4 *Let $S = \langle T, V, P, C, \omega\rangle$ and $S' = \langle T', V', P', C, \omega'\rangle$ be two Cartesian product-compatible SPOs. Let $S'' = \langle T'', V'', P'', C, \omega''\rangle = S \times S'$. Then P'' is the correct joint probability distribution of random variables in V and V' under the assumption of independence between them, given distributions P of V and P' of V'.*

THEOREM 5 *Let $S = \langle T, V, P, C, \omega\rangle$ and $S' = \langle T', V', P', C, \omega'\rangle$ be two join-compatible SPOs. Let $S'' = \langle T'', V'', P'', C, \omega''\rangle = S \rtimes S'$ and $S''' = \langle T''', V''', P''', C, \omega'''\rangle = S \ltimes S'$. Then P'' and P''' are the correct joint probability distributions of random variables in V and V' under the assumption of independence between them, given distributions P of V and P' of V'.*

THEOREM 6 *Let S and S' be two join-compatible SPOs. The left join $S \ltimes S'$ and right join $S \rtimes S'$ are equivalent if and only if the two SPOs are join-consistent.*

5. Semistructured Probabilistic DBMS

In this section we describe in detail the design and implementation of the database management system for SPOs.

5.1. Representation of SPOs

One consideration in the design of the SPDBMS was that it could take over the data management routine from complex AI applications dealing with uncertain data. Applications such as support of Bayes net construction typically consist of different components, some of which extract and/or elicit the probability tables while others support the construction and further use of Bayes nets given the data. If SPDBMS takes on the role of the data backbone of such an application, representation of SPOs for communication between different components of the system becomes important. The representation mechanism must be transparent and easy to use by diverse applications.

Extensible Markup Language (XML) [3] provides us the benefit of using clear APIs for parsing and processing data, together with open source software implementing these tasks, relieving the SPDBMS from the need to do its own syntactic parsing. This makes SPO data encoded in XML easy to pass from component to component. We represent SPOs in XML using a markup meta-language that we call SPO-ML. We use the names of random variables and context attributes as element names in the markup language. Thus, the actual DTD/XML schema of the markup language depends on the application domain, namely the pair $\langle \mathcal{V}, \mathcal{R}\rangle$. SPO-ML simply represents the general markup rules for any domain.

For example, consider an application domain with the universe $\mathcal{V}$ of random variables $\{$v1, v2, …, vn$\}$ and a collection of context attributes $\mathcal{R} = (A1, \ldots, Ak)$. We construct the appropriate markup language as shown on the template DTD in Figure 9 [4].

<!DOCTYPE **spo** [
<!ELEMENT **spo** (**context**?, **table**, **conditional**?)>
<!ELEMENT **context** (($A_1|A_2|...|A_i|...|A_k$)*)>
<!ELEMENT **table** (**row+**)>
<!ELEMENT **conditional** ($v_1?, v_2?, ...v_j?, ..., v_n?$)>
<!ELEMENT **row** ($v_1?, v_2?, ...v_j?, ..., v_n?,$ **P**)>
<!ELEMENT A_i (#PCDATA)>
<!ELEMENT v_j (#PCDATA)>
<!ELEMENT **P** (#PCDATA)>
<!ATTLIST **spo**
  path #PCDATA #REQUIRED>
]>

Figure 9. XML DTD template for SPOs in the specified application domain.

ω: S

race : Asian

DA	LY	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	0.0
F	B	0.01
F	C	0.02
F	F	0.04

DR $\in$ { A, B }

```
<?xml version="1.0"?>
<spo path = "S">
  <context>
    <race> Asian </race>
  </context>
  <table>
    <row> <DA>A</DA> <LY>A</LY> <P>0.09 </P> </row>
    <row> <DA>A</DA> <LY>B</LY> <P>0.12 </P> </row>
    <row> <DA>A</DA> <LY>C</LY> <P>0.03 </P> </row>
    <row> <DA>A</DA> <LY>F</LY> <P>0.005</P> </row>
     ...                ...                       ...
    <row> <DA>F</DA> <LY>A</LY> <P>0.0  </P> </row>
    <row> <DA>F</DA> <LY>B</LY> <P>0.01 </P> </row>
    <row> <DA>F</DA> <LY>C</LY> <P>0.02 </P> </row>
    <row> <DA>F</DA> <LY>F</LY> <P>0.04 </P> </row>
  </table>
  <conditional>
    <DR> {A B} </DR>
  </conditional>
</spo>
```

Figure 10. A typical SPO object for risk level on Driver Age and License Years, and its XML representation.

Figure 10 shows an SPO and its encoding in SPO-ML. The top layer of the XML encoding of the SPO consists of three elements: <context>, <table> and <conditional>. The path is represented as an attribute for the <spo> element. The content of the context and conditional parts are straightforward. The probability table is modeled as a collection of rows, each of which consists of a sequence of random variables with values and the corresponding probability.

Semistructured Probabilistic Objects are complex structures and not all their properties can be captured by XML validity checks. An SPO representation in SPO-ML should satisfy the following extra validity constraints. First, all <row> elements inside the <table> elements have to have exactly the same sequence of participating random variables. Second, the set of random variable elements inside <table> and the set of random variable elements inside <conditional> must be disjoint. Finally, the content of <P> elements inside <row> elements is expected to be real numbers between 0 and 1, and their sum must be less than or equal to 1. These additional constraints mean that validation on an XML representation of an SPO is a two-step process: first the XML is validated against the appropriate DTD/XML schema, and then the additional constraints are verified. While the validity of the SPO-ML documents is checked by a validating parser, these extra checks are performed by the SPDBMS itself.

5.2. *Architecture of SPDBMS*

We have implemented a prototype semistructured probabilistic database system on top of a RDBMS in Java, JDK1.3. Figure 11 depicts the overall architecture of our system. The core of the system is the SPDBMS application server which processes query requests from a variety of client applications. The application server provides a JDBC-like API, through which client applications can send standard database management instructions, such as CREATE DATABASE, DROP DATABASE, CREATE SP-RELATION, DROP SP-RELATION, INSERT INTO SP-RELATION, DELETE FROM SP-RELATION, as well as SP-Algebra queries to the server.

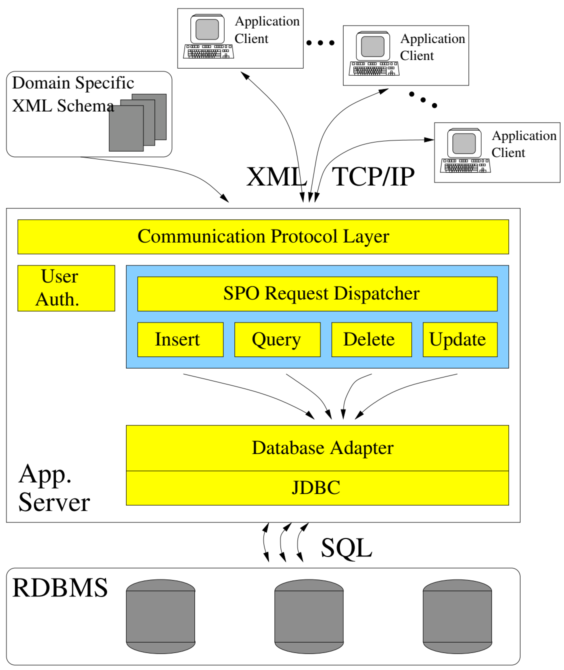

Figure 11. The overall architecture of SPDBMS.

5.3. Mapping SPOs to relational tables

A relational database system has been used as a backend to store SPO-ML encoded data by mapping the XML schema onto a set of relational tables. Numerous techniques for converting XML documents into relational databases exist [26, 13, 17, 9]. As shown in [26], none of the proposed translation schemes is monotonically better than all the others. These schemes are proposed for storage of arbitrary XML with unknown structure. In the case of storing SPO-ML <spo> elements in a relational database, we can take advantage of our knowledge of the general structure of these elements when designing the translation mechanism. This consideration leads us to adopt a translation scheme, described below, that is specific to the structure of SPO-ML objects instead of a generic mechanism.

The SPO-ML - to relational database translation works as follows. SPOs are stored in a relational database with the following schema:

RELATION (rid integer, name varchar, schema varchar) contains SP-relation level information. It connects all other tables by using the table naming convention that every table uses the unique identifier of the corresponding SP-relation *rid* as a prefix for its name. The attributes *name* and *schema* represent the SP relation name and the corresponding schema URL, respectively.

rid_SPO (id integer, path varchar, head varchar, numvar integer) contains SPO level information. The association between this table and other tables is established by the unique identifier *id* of an SPO. The attribute *head* stores the prolog of an XML document and *numvar* stores the number of participating random variables in an SPO.

rid_SPO_CONS (id integer, type char, elemname varchar, elemvalue varchar, idref varchar) contains all the information about SPO context and conditional. The attribute *id* is a foreign key, and *type* tells whether it's a context or conditional. The attributes *elemname* and *elemvalue* give the element name and element value.

rid_SPO_VAR (id integer, position integer, varname varchar) contains all the information about the participating random variables of SPOs The attributes *position* and *varname* represent a pair of position and variable name.

rid_SPO_num (id integer, var_1 char, ... , var_num char, p decimal)(num is a variable, which equals the number of participating variables in a particular SPO. num may vary from 1 to $|\mathcal{V}|$.) contains all the information of the probability tables for SPOs which have *num* participating random variables. The attribute *p* stores the probability value.

In order to improve data integrity and query performance, we created primary keys and foreign keys, such as primary key *rid* for relation RELATION, primary key *id* for relation *rid*_SPO and foreign keys *id* for all other relations. We also created indices for the last three type of relations, for instance, multicolumn index on (*id*, *elemname*) for relation *rid*_SPO_CONS, multicolumn index on (*id*, *varname*) for relation *rid*_SPO_VAR and multicolumn index on (*id*, *var*_1,..., *var*_num) for relation *rid*_SPO_num.

The database system stores SPOs from each SP-relation in a separate set of relational tables. The CREATE algorithm starts by storing the SP-relation name, the path and the schema url, generates a unique identifier *rid* for the SP-relation. It also creates all the empty tables with the schema defined above and associates them with the SP-relation. In order to store SPOs in an SP-relation, the SPOs must be parsed to a Document Object Model (DOM) tree and decomposed into four components, Head, Path, Context, Table and Conditional, based on a predefined schema template. The INSERT algorithm gets a unique SPO object identifier *id*, then stores the XML prolog information, the path and the number of participating random variables in the SPO in the rid_SPO table. It stores SPO context and conditional in the table rid_SPO_CONS, the probability table in the table rid_SPO_num and participating random variables in the table rid_SPO_VAR, respectively. Figure 12 shows the resulting tables after storing the SPO defined in Figure 3 in an SP-relation.

1_SPO_2

id	var_1	var_2	P
2	A	A	0.09
2	A	B	0.12
2	A	C	0.03
2	A	F	0.005
2	B	A	0.12
2	B	B	0.16
2	B	C	0.13
2	B	F	0.01
2	C	A	0.03
2	C	B	0.08
2	C	C	0.11
2	C	F	0.045
2	F	A	0.0
2	F	B	0.01
2	F	C	0.02
2	F	F	0.04

RELATION

rid	name	schema
1	First	schema_url

1_SPO

id	path	head	numvar
2	S	null	2

1_SPO_VAR

id	position	varname
2	1	DA
2	2	LY

1_SPO_CONS

id	type	elemname	elemvalue	idref
2	cont	race	Asian	null
2	cond	DR	{A,B}	null

Figure 12. internal representation for the SPO defined in Figure 3.

5.4. *Querying the SPDBMS*

The SP-Algebra operations described in previous sections have been implemented. The query language allows us to navigate through the entire database with structured queries, including any combination of SELECTION, PROJECTION, CONDITIONALIZATION, CARTESIAN PRODUCT and JOIN. In the current implementation structured queries are first parsed and then transformed into a query tree [5]. Each internal node in the resulting parse tree is an SP-Algebra operator and each of the leaves is an SP-relation. Each operator is translated in a straightforward manner into a sequence of corresponding SQL statements that can be executed by the underlying RDBMS.

This, however, is not enough for some SP-Algebra queries. Conditionalization, projection, Cartesian product and join change the probability tables in the results according to the semantics of each operation. These computations are performed at the SPDBMS server during the special *postprocessing* stages of the query processing. Postprocessing also includes the assembly of the resulting XML document.

Space limitations prevent us from describing query translations for all SP-Algebra queries. Here we give two examples of probabilistic queries, illustrating how to map from an SP-Algebra query to a set of SQL statements; other translations can be found in [27]. First, consider **selection on probabilities**. Given a selection condition P op x and an SP-relation $\mathcal{S} = \{S_1, \ldots, S_n\}$, this operation returns SPOs that have at least one row with probability that satisfies the selection condition. Selection preserves the context, participating random variables and conditionals in the original SPOs, but returns only those rows of the probability table satisfying the selection condition. Consider a sample query $\sigma_{p>0.1}(\mathcal{S})$. Figure 13 shows the sequence of SQL statements needed in order to evaluate this query.

```
step 1. Get the SPO ID list based
        on given probability value:
SELECT DISTINCT id
FROM rid_SPO_1
WHERE p > 0.1
...
UNION ALL
SELECT DISTINCT id
FROM rid_SPO_i
WHERE p > 0.1
...
UNION ALL
SELECT DISTINCT id
FROM rid_SPO_max
WHERE p > 0.1

step 2. Get the variable list for
        each SPO in the ID list:
SELECT id, varname, position
FROM rid_SPO_VAR
WHERE id IN {ID list}

step 3. Retrieve context/conditional:
SELECT id, elemname, elemvalue, idref
FROM rid_SPO_CONS
WHERE id IN {ID list}

step 4. Retrieve probability table
SELECT *
FROM rid_SPO_1
WHERE id IN {ID list}
      AND p > 0.1
...
UNION ALL
SELECT *
FROM rid_SPO_i
WHERE id IN {ID list}
      AND p > 0.1
...
UNION ALL
SELECT *
FROM rid_SPO_max
WHERE id IN {ID list}
      AND p > 0.1
```

Figure 13. Steps to evaluate the selection query

Our second example is the operation of **conditionalization**. This operation computes conditional probability distributions and thus, is specific to *probabilistic* algebras. Given the constraint $v = a$ and an SP-relation $\mathcal{S} = \{S_1, \ldots, S_n\}$, conditionalization $\mu_{v=a}$ first selects SPOs in which v is a participating random variable. For each selected SPO, conditionalization preserves the context, conditions the joint probability distribution, replaces the probability table with a new conditional probability distribution over the remaining random variables, and adds the condition $(v, \{a\})$ to the conditional part of each of the resulting SPOs. Consider a sample query $\mu_{DA='A'}(\mathcal{S})$. In order to perform the conditionalization operation, first a sequence of SQL statements, as shown in Figure 14, need to be issued against the underlying RDBMS to retrieve all the SPOs satisfying the condition. This, however, is not enough for the conditionalization operation since this operation changes the probability tables in the results according to the semantics of the operation. These computations are performed by the *postprocessing* process, and the postprocesing algorithm is shown in Figure 15.

```
step 1. Get the SPO ID list based
        on the variable name(s):
SELECT DISTINCT id
FROM rid_SPO_VAR
WHERE varname = 'DA'

step 2. Get the variable list for
        each SPO in the ID list:
SELECT id, varname, position
FROM rid_SPO_VAR
WHERE id IN {ID list}

step 3. Retrieve context/conditional:
SELECT id, elemname, elemvalue, idref
FROM rid_SPO_CONS
WHERE id IN {ID list}

step 4. Retrieve probability table
SELECT id, {var_i list}, P
FROM rid_SPO_1
WHERE id IN {ID list}
      AND var_position = 'A'
...
UNION ALL
SELECT id, {var_j list}, P
FROM rid_SPO_i
WHERE id IN {ID list}
      AND var_position = 'A'
...
UNION ALL
SELECT id, {var_k list}, P
FROM rid_SPO_max
WHERE id IN {ID list}
      AND var_position = 'A'
```

Figure 14. Steps to evaluate the conditionalization query

```
step 1. Store the retrieved SPOs in predefined data structures;
step 2. Remove the entire column for attribute DA;
step 3. Compute the sum of all the probabilities;
step 4. Divide all the probabilities by the computed sum;
step 5. Add a new condition (i.e. DA = 'A') in the conditional part;
step 6. Assemble the final SPOs into an XML document.
```

Figure 15. Postprocessing algorithm for the conditionalization query

5.5. *Experimental results*

In this section we present the results of tests conducted with the prototype system. The current system uses Oracle8i as the RDBMS back-end. To avoid network delays during tests, both the application server and Oracle DB server were running on the same machine, a 440 MHz Sun Ultra 10 running Solaris OS with 1GB of main memory, and the timing was done on the server side.

In order to ensure consistency, each experiment consists of 20 runs, and each point on a graph represents the average running time for the 20 runs. We also restarted the application server for each experiment to minimize the time difference consumed by garbage collection. Most test data sets used in the experiments are generated randomly by a custom data generator[6]. However, for Cartesian product and join, we generated specific data sets in order to control the selectivity for each query. Each data set was generated based on the following three parameters: number of SPOs per SP-relation, number of participating random variables in an SPO, and size of the domain of participating random variables. The first parameter affects the number of objects to be stored in the database while the other two affect the size of individual objects. Throughout the experiments, we used a fixed number of context and conditional elements in a single SPO. So the last two parameters specify the internal structure of each SPO and consequently the size of each SPO. Table 2 shows some typical data sets with corresponding file size and number of tuples in the underlying Oracle database.

Table 2. File size and number of tuples in Oracle database for typical data sets.

number of SPOs	1,000	1,000	1,000	10,000	10,000	10,000
number of variables	2	4	2	2	4	2
size of domain	2	2	4	2	2	4
size of original XML file (MB)	0.38	1.64	1.10	3.81	16.4	11.0
number of tuples in Oracle DB	10,000	24,000	22,000	100,000	240,000	220,000

We examined the running time for each type of atomic SP-Algebra query. Most queries are generated randomly at runtime by a custom query generator[6] in the client application running on another machine. We have collected both the total running time and the time consumed by the Oracle DB server for executing SQL statements. The Oracle server consumes 75 - 95% of the total execution time for most queries, and the percentage increases with the size of the XML files. A typical case is shown in Table 3.

Table 3. Time distribution between Oracle and postprocessing for data set with 3 variables and 10,000 SPOs and domain size of 2.

Test type	Total time/sec	Oracle time/sec	Postprocessing/sec	%Oracle
Select on context	1.193	1.053	0.140	88
Select on conditional	0.955	0.845	0.110	88
Select on variable	1.081	0.959	0.122	88
Select on table	0.736	0.661	0.075	89
Project on conditional	1.080	0.958	0.122	88
Project on variable	0.502	0.471	0.031	93
Conditionalization	0.769	0.619	0.150	80

To study the effects of the number of SPOs in an SP-relation on the query running time, different experiments were conducted with the number of SPOs varying from 10 to 10,000. The results are plotted in Figure 16. It can be observed that all types of unary SP-Algebra queries scale well as the size of SP-relations increases: the running time increases sublinearly with the number of SPOs for large SP-relations, but at a much slower rate for SP-relations of small size. In Figure 17, the effects of domain size for participating variables are shown. Notice that the number of tuples considered grows polynomially (i.e.

quadratically in this case, the number of variables equals 2) with the size of the domain. The running time increases with the size of domain, but not as quickly as does the size of the XML files.

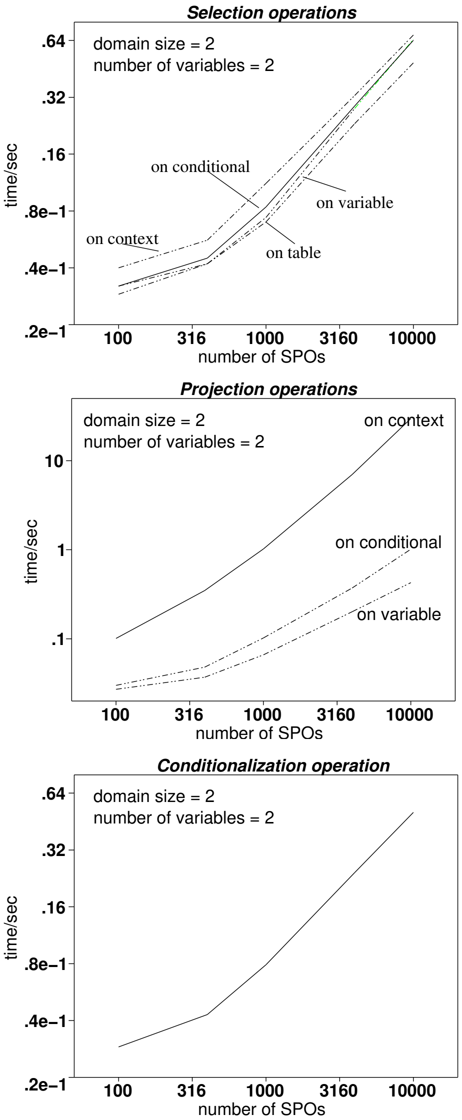

Figure 16. Effect of number of SPOs in an SP-relation.

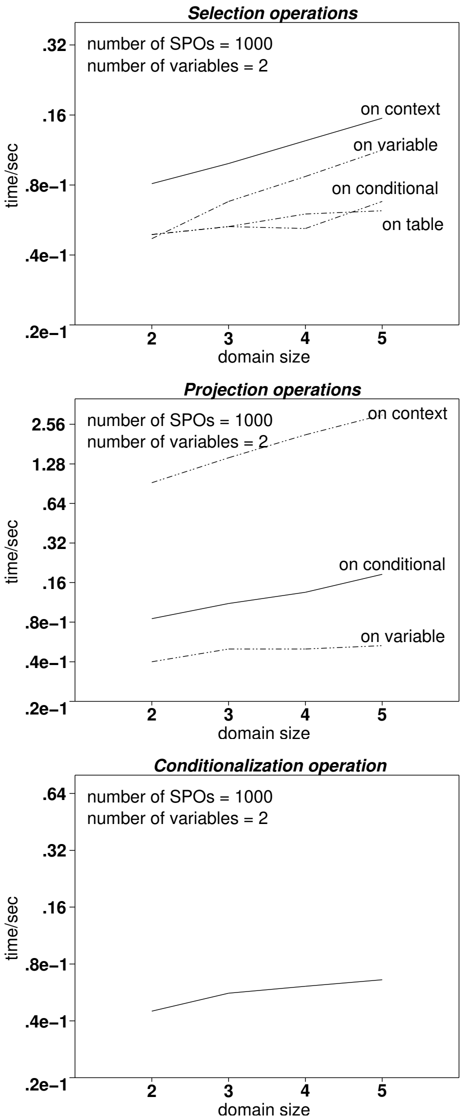

Figure 17. Effect of domain size of participating random variables.

Figure 18 shows the effects of the number of variables in SPOs on the time for the conditionalization operation. The effect of running time on the size of the SP-relation and query selectivity for selection on probability is shown in Figure 19. We can see that the running time increases with selectivity and at faster rate in lower selectivity. It also increases linearly with the number of SPOs. Finally, Figure 20 shows the dependence of running time on the size of the SP-relation and query selectivity for the Cartesian product operation. The graph shows that the running time for Cartesian product increases with the number of SPOs at a nearly quadratic rate, and also increases with the selectivity, but at a much slower rate. One reason is that the number of SPOs output increases quadratically with the number of SPOs in the initial SP-relations. The same effect can be seen for the join operation, as shown in figure 21.

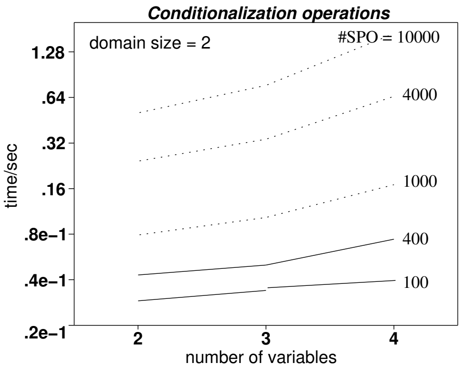

Figure 18. Effect of number of variables on conditionalization operation.

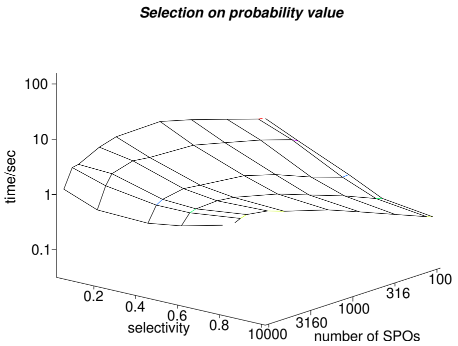

Figure 19. Effect of number of SPOs and selectivity on selection query on probability.

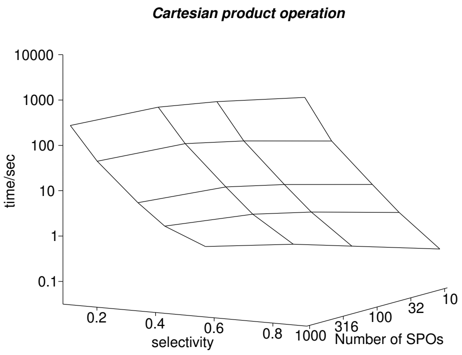

Figure 20. Effect of number of SPOs and selectivity on Cartesian Product.

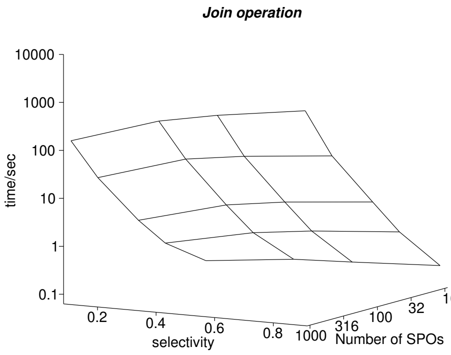

Figure 21. Effect of number of SPOs and selectivity on join operation.

5.6. Advantages and drawbacks

The algorithms for storing and querying SPOs in a structure-oriented way are completely independent of the underlying RDBMS. All features specific to a RDBMS are implemented in a database adapter class, so the system can port to any relational database with little modification. The mapping of SPOs onto sets of relational tables makes queries efficient, especially queries on specific parts of SPOs. No information loss occurs during the decomposition. However, in order to ensure that the probability information is stored and manipulated correctly, the current decomposition algorithm produces five predefined components for each SPO to be inserted into the relational database. Thus, only XML objects that conform to the predefined SPO schema template can be stored in the database.

6. Related Work

While modeling and managing uncertain information has received considerable attention in the last two decades, most of that work has been in the context of probabilistic relational databases. Different probabilistic relational models have been developed. Cavallo and Pittarelli [5] first outlined a theory of probabilistic relational databases which incorporates probability into relational data. The probabilistic system was defined as a four-tuple $P = (V, \Delta, dom, p)$, where V is a non-empty set of distinct attributes, Δ is a non-empty set of domains of attributes, dom : V $\longrightarrow$ Δ is a function that associates a domain with each attribute, and p : dom(V) $\longrightarrow$ $[0, 1]$ is a probability distribution over V. Their data model requires that the probabilities for all the tuples in a relation must add up to exactly 1. As a result, separate relations are needed to represent different objects. They focused their work on information content, functional dependency and multivalued dependency. They defined only two probabilistic relational operations, namely projection and join, in this context. Pittarelli [23] extended the probabilistic algebra defined in [5] to include some new operators, e.g. the pooling operator, which combines estimates from different sources into a single distribution. A common approach is to use linear pooling which computes a weighted average of different estimates. We are investigating techniques of data fusion which are similar to the idea of pooling.

Barbará, Garcia-Molina and Porter [2] presented a non-1NF probabilistic data model as an extension of the relational model. In their model, relations have deterministic keys, and tuples with different keys represent real world entities. All the non-key attributes describe the properties of the entities and may be deterministic or stochastic, and independent or interdependent. Probabilities are associated with the values of stochastic attributes, and the interdependent relationship indicates that the attributes involved are jointly distributed ones. Besides the basic relational operators, they also introduced a new set of operators to illustrate the various possibilities. For example, the STOCHASTIC operator takes as input a deterministic relation (one where all attributes are deterministic) and returns a probabilistic relation according to a specified probabilistic schema. The DISCRETE operator goes the opposite direction. It takes as input a probabilistic relation and returns a deterministic expected value relation.

Dey and Sarkar [10] provided a probabilistic database framework with relations abiding by first normal form (1NF). Unlike Barbará, et al. [2], they assigned probabilities to tuples, instead of individual attributes, in terms of joint probability distribution. they required the sum of all probabilities associated with a key value to be no more than 1. They provided a closed form query algebra and first introduced the conditionalization operation in the context of a probabilistic model. Later they proposed a non-procedural probabilistic query language called PSQL [11] as an extension of the SQL language.

Based on a first-order probabilistic logic language proposed by Halpern [14], Zimányi [29] formalized a relational model to represent probabilistic information. The data model is similar to [10]. Zimányi also provided a complete method for evaluating queries in probabilistic theories. Lakshmanan, et al.[19] proposed axioms characterizing reasonable probabilistic conjunction and disjunction strategies. They first implemented a relational probabilistic database system called ProbView.

Instead of modeling uncertain information with relational models, Kornatzky and Shimony [18] developed a probabilistic object-oriented model to represent uncertain information based on a probabilistic calculus. Uncertainty in the values of attributes was represented by probabilities. One limitation is that they assume that all events involved are independent. Eiter, et al.[12] extended the work in [18] by proposing an algebra for the probabilistic object bases. Unlike the previous work, their algebra allows users to specify dependencies between events.

All these approaches above are extensions to either relational databases or object databases, with the limitations inherent in each. The probabilistic object (e.g. as described in [12]) represents a single real world entity with uncertain attribute values. In our case, an SPO represents a probability distribution of one or more random variables. Our work combines and extends the ideas contained in these papers and applies them to a semistructured data model, which provides us with the benefit of schema flexibility. For instance, this model provides additional context information, providing general information for the probability distribution and conditional information, making it possible to represent conditional probability distributions.

There are two approaches to semistructured probabilistic data management that are closely related to ours: the ProTDB [21] and the PIXml [16] frameworks. Nierman, et al.[21] extends the XML data model by associating a probability to each element with the modification of regular non-probabilistic DTDs. They provided two ways of modifying non-probabilistic DTDs. One is to introduce to every element a probability attribute Prob to specify the probability of the particular element existing at the specific location of the XML document. The other is to attach a new subelement called Dist to each element, which makes it possible to represent probability distributions. One drawback is that in their model probabilities in an ancestor-descendant chain are related probabilistically, meaning that probabilities in the document are always conditional probability. All other probabilities are assumed to be independent. Hung, Getoor and Subrahmanian [16] proposed a probabilistic interval XML data model, PIXml. They provided two types of semantics for uncertain data, along with connections between the two. The global interpretation is a distribution over an entire XML document, while the local interpretation specifies an object probability function for each non-leaf object. They also proposed a path expression-based query language to access stored information. This approach overcomes some drawbacks presented in [21], but does not provide a convenient way to represent joint probability distributions.

Our approach is different from theirs in that we define probability distributions over a set of random variables, along with contextual information and conditional information. The context provides general information for the probability distribution, and the conditional specifies conditions for the probability distribution. Also, our framework provides a comprehensive query algebra to efficiently query the semistructured probabilistic database. The algebra presented here works on the semistructured data irrespective of the format in which it is actually stored. The semistructured probabilistic algebra presented here has no data format-specific syntax.

In this paper, all probability distributions considered are point probability distributions. It is reasonable to ask whether a similar framework can be designed to handle imprecise probabilities. We have designed a sister database framework for handling imprecise probabilities as represented by probability intervals, in SPOs [28]. Because interval probability distributions provide a richer framework than point probabilities [6], certain restrictions in the work described here are removed, such as assumptions about independence.

7. Conclusions and Future Work

In this paper, we presented a semistructured probabilistic database framework for storing and managing probabilistic information. The SPO data mode has been defined to represent probabilistic distributions over arbitrary sets of random variables, along with additional information applicable to the probabilistic distributions. This construction allows us to specify general information about a probabilistic distribution and express conditional probabilistic distributions by specifying conditions associated with the probabilistic distribution. We described the pilot implementation of this data model. We also reported a performance evaluation of the SPDBMS for different types of SP-Algebra queries.

There are three foci of our ongoing work: (i) implementation of a query optimizer for the current semistructured probabilistic DBMS; (ii) extension of the data model and the algebra to handle probabilistic information with broader structures, and (iii) study of data fusion and conflict resolution problems that arise in this framework.

Acknowledgments

This work was partially supported by NSF grants CCR-0100040, ITR-0325063, and ITR-0219924. We'd like to thank the anonymous reviewers of our conference papers, 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.

Notes

1. 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.
2. The list of variables is also used in the projection onto the participating random variables. The syntax $c : \mathcal{F}$ is chosen to distinguish between the two types of projection operation.
3. In general, given two events a and b and their point probabilities $p(a)$ and $p(b)$, the probability $p(a \wedge b)$ of their conjunction lies in the interval $[\max(0, p(a) + p(b) - 1), \min(p(a), p(b))]$. One can obtain a point probability for $p(a \wedge b)$ only if a specific relationship between a and b, such as independence, positive or negative correlation is known to exist between them, or assumed.

4. The DTD representation of SPO-ML is chosen here for its simplicity and succinctness. We also maintain and use the corresponding XML schemas.
5. Work on query optimization in SPDBMS is underway, but the current version of the SPDMBS server does not have this feature.
6. Both the SPO data generator and the SP-Algebra query generator utilize a linear congruential pseudo-random number generator, which comes with Sun JDK1.3 package.

References

1. S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web: From Relations to Semistructured Data and XML.* Morgan Kaufmann, 1999.
2. D. Barbará, H. Garcia-Molina, and D. Porter. The management of probabilistic data. *IEEE Trans. on Knowledge and Data Engineering*, 4:487 – 502, 1992.
3. T. Bray, J. Paoli, and C.M. Spreberg-McQueen (Eds.). Extensible markup language (XML) 1.0. *World Wide Web Consortium Recommendation*, 19980210, 1998.
4. P. Buneman. Semistructured data. In *Proc. PODS'97*, pages 117 – 121, 1997.
5. R. Cavallo and M. Pittarelli. The theory of probabilistic databases. In *Proc. VLDB'87*, pages 71 – 81, 1987.
6. Luis M. de Campos, Juan F. Huete, and Serafin Moral. Probability intervals: A tool for uncertain reasoning. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 2(2):167 – 196, 1994.
7. A. Dekhtyar, J. Goldsmith, and S. Hawkes. Semistructured probabilistic databases. In *Proc. Statistical and Scientific Database Management Systems*, 2001.
8. A. Dekhtyar and V.S. Subrahmanian. Hybrid probabilistic logic programs. *Journal of Logic Programming*, 43(3):187 – 250, 2000.
9. A. Deutsch, M. Fernandez, and D. Suciu. Storing semi-structured data using STORED. In *Proc., ACM SIGMOD*, pages 431 – 442, 1999.
10. D. Dey and S. Sarkar. A probabilistic relational model and algebra. *ACM Transactions on Database Systems*, 21(3):339 – 369, 1996.
11. D. Dey and S. Sarkar. PSQL: A query language for probabilistic relational data. *Data and Knowledge Engineering*, 28:107 – 120, 1998.
12. T. Eiter, J. Lu, T. Lukasiewicz, and V.S. Subrahmanian. Probabilistic object bases. *ACM Transactions on Database Systems*, 2001.
13. D. Florescu and D. Kossmann. A performance evaluation of alternative mapping schemes for storing xml data in a relational database. Technical Report 3680, INRIA Technical Report, 1999.
14. J. Halpern. An analysis of first-order logics of probability. *Artificial Intelligence*, 46(3):311 – 350, 1990.
15. S. Hawkes and A. Dekhtyar. Designing Markup Languages for Probabilistic Information, University of Kentucky Tech. Report, TR 319-01, 2001.
16. E. Hung, Lise Getoor, and V.S. Subrahmanian. Probabilistic interval XML. In *Proc. of the Ninth International Conference on Database Theory*, 2003.
17. C.-Ch. Kanne and G. Moerkotte. Efficient strorage of XML data. In *Proc., ICDE*, page 198, 2000.
18. E. Kornatzky and S.E. Shimony. A probabilistic object data model. *Data and Knowledge Engineering*, 12:143 – 166, 1994.
19. V.S. Lakshmanan, N. Leone, R. Ross, and V.S. Subrahmanian. Probview: A flexible probabilistic database system. *ACM Transactions on Database Systems*, 22(3):419 – 469, 1997.
20. R. Ng and V.S. Subrahmanian. Probabilistic logic programming. *Information and Computation*, 101(2):150 – 201, 1993.
21. A. Nierman and H. V. Jagadish. ProTDB: Probabilistic data in XML. In *Proc. of the 28th VLDB Conference*, 2002.
22. J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.
23. Michael Pittarelli. An algebra for probabilistic databases. *IEEE Transaction on Knowledge and Data Engineering*, 6(2):293 – 303, 1994.
24. S.J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.
25. D. Suciu. Semistructured data and XML. In *Proc. 5th. Intl. Conf. on Foundation of Data Organization*, pages 1 – 12, 1998.
26. F. Tian, D.J. DeWitt, J. Chen, and C. Zhang. The design and performance evaluation of alternative xml storage strategies. *SIGMOD Record*, 31(1):5 – 10, 2002.

27. W. Zhao, A. Dekhtyar, and J. Goldsmith. Representing probabilistic information in XML. Technical Report 770-03, Department of Computer Science, University of Kentucky, 2003.
28. Wenzhong Zhao, Alex Dekhtyar, and Judy Goldsmith. Databases for interval probabilities. *International Journal of Intelligent Systems*, 2004. To appear.
29. Esteban Zimányi. Query evaluation in probabilistic relational databases. *Theoretical Computer Science*, 171:179 – 219, 1997.

Appendix

Proof: Proof of Theorem 1
Here we prove that different selection operations commute.

Let c and c' be two atomic selection conditions. There are 5 types of atomic selection conditions, namely *context*, *participation*, *conditional*, *table* and *probability*. Selection on *context*, *participation* or *conditional* will result in entire SPOs being selected, while selection on *table* or *probability* will select only parts of the relevant SPOs. We could partition the conditions into two groups,

- Group I, containing *context*, *participation* and *conditional* conditions, and
- Group II, containing *table* and *probability* conditions.

First we prove that $\sigma_c(\sigma_{c'}(S)) = \sigma_{c'}(\sigma_c(S))$ for a single SPO S.

Case 1. Both conditions c and c' are in Group I.

There are three possible combinations for whether each condition is satisfied:
a) S satisfies c but not c', or
b) S satisfies c' but not c, or
c) S satisfies both c and c', or
d) S does not satisfy either c or c'.

By the definition of selection on atomic selection conditions in Group I, we know selection on these conditions will result in the entire SPO being selected, or none of it.

For case (a), since S does not satisfy c', $\sigma_{c'}(S)$ returns empty and subsequently $\sigma_c(\sigma_{c'}(S))$ will return empty. Since $\sigma_c(S)$ returns S, we see that $\sigma_{c'}(\sigma_c(S)) = \sigma_{c'}(S)$ will also return empty for the same reason. Thus, $\sigma_c(\sigma_{c'}(S)) = \sigma_{c'}(\sigma_c(S))$ holds for case (a). The same applies to case (b). Similarly, for case (d). For case (c), $\sigma_c(\sigma_{c'}(S)) = \sigma_c(S)$ returns S, and $\sigma_{c'}(\sigma_c(S)) = \sigma_{c'}(S)$ returns S too. Thus, $\sigma_c(\sigma_{c'}(S)) = \sigma_{c'}(\sigma_c(S))$ holds for case (c).

So $\sigma_c(\sigma_{c'}(S)) = \sigma_{c'}(\sigma_c(S))$ holds for all these cases.

Case 2. Condition c is in Group I and condition c' is in Group II.

There are only two possible combinations for whether each condition is satisfied, assuming that condition c' is always partially satisfied:
a) S does not satisfy c, or
b) S satisfies c.

By the definition of selection on atomic selection conditions in both Group I and Group II, we know selection on conditions in Group I will result in the entire SPO being selected or not, while selection on conditions in Group II will preserve all the context, participating random variables and conditionals in the original SPO, but produce only a part of the probability table.

Let $\sigma_{c'}(S) = S'$, where S' has the part of the probability table that satisfies the condition c' and retains all the context, participating random variables and conditionals in S.

For case (a), $\sigma_c(\sigma_{c'}(S)) = \sigma_c(S')$ will return empty since S' does not satisfy the condition c either. Since $\sigma_c(S)$ returns empty, subsequently $\sigma_{c'}(\sigma_c(S))$ will also return empty. This proves $\sigma_c(\sigma_{c'}(S)) = \sigma_{c'}(\sigma_c(S))$ for case (a). For case (b), $\sigma_c(\sigma_{c'}(S)) = \sigma_c(S')$ will return S', since S' should also satisfy the condition c. Since $\sigma_c(S)$ returns S, so $\sigma_{c'}(\sigma_c(S)) = \sigma_{c'}(S)$ will also return S'. This proves that $\sigma_c(\sigma_{c'}(S)) = \sigma_{c'}(\sigma_c(S))$ holds for case (b).

So $\sigma_c(\sigma_{c'}(S)) = \sigma_{c'}(\sigma_c(S))$ holds for both cases.

Case 3. Both c and c' are conditions in Group II.

First we prove $\sigma_c(\sigma_{c'}(S)) = \sigma_{c'}(\sigma_c(S))$ for a single SPO S. Assume that both conditions c and c' are partially satisfied by S. By the definition of selection on atomic selection conditions in Group II, we know that selection on these conditions will result in part of the probability table and will preserve all the context, participating random variables and conditionals in the original SPO. In other words, all the components in the original SPO except the probability table will be preserved.

Let $S = \langle T, V, P, C\rangle$. Then $S' = \sigma_c(\sigma_{c'}(S)) = \langle T, V, P', C\rangle$ and $S'' = \sigma_{c'}(\sigma_c(S)) = \langle T, V, P'', C\rangle$ with $P' = \varrho_c(\varrho_{c'}(P)$ and $P'' = \varrho_{c'}(\varrho_c(P)$, where ϱ is the relational selection operator. Since the relational selection operator ϱ is commutative, $\varrho_c(\varrho_{c'}(P) = \varrho_{c'}(\varrho_c(P)$. Therefore we have $P' = P''$ or $S' = S''$. So $\sigma_c(\sigma_{c'}(S)) = \sigma_{c'}(\sigma_c(S))$ holds for this case.

Now let an SP-relation $\mathcal{S} = \cup_{S\in\mathcal{S}} S$. Since the union operator is commutative, i.e.
$\sigma_{c'}(\sigma_c(\mathcal{S})) = \sigma_{c'}(\sigma_c(\cup_{S\in\mathcal{S}}(S))) = \cup_{S\in\mathcal{S}}(\sigma_{c'}(\sigma_c(S)))$ and
$\sigma_c(\sigma_{c'}(\mathcal{S})) = \sigma_c(\sigma_{c'}(\cup_{S\in\mathcal{S}}(S))) = \cup_{S\in\mathcal{S}}(\sigma_c(\sigma_{c'}(S)))$.

This proves $\sigma_c(\sigma_{c'}(\mathcal{S})) = \sigma_{c'}(\sigma_c(\mathcal{S}))$.

■

Proof: Proof of Theorem 2
Let $S = \langle T, V, P, C, \omega\rangle$ be a Type I SPO and $\mathcal{L}_V \subset V$. We prove that the projection operation correctly computes the marginal probability distribution.

Let $P(\bar{x}, \bar{y})$ be a probability distribution with $(\bar{x}, \bar{y}) \in dom(V)$. By the definition of projection in Section 4.3, we know that $P' : dom(\mathcal{L}_V) \longrightarrow [0, 1]$ and for each $\bar{x} \in dom(\mathcal{L}_V)$,

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

Note that this sum is exactly the marginal probability, for any $\bar{x} \in dom(\mathcal{L}_V)$.

■

Proof: Proof of Theorem 3
Let $S = \langle T, V, P, C, \omega \rangle$ be a Type I SPO and let c be a conditional selection condition involving variable $v \in V$. We prove that the conditionalization computes the correct conditional probability distribution.

Let $P(\bar{x}, \bar{y})$ be a probability distribution with $(\bar{x}, \bar{y}) \in dom(V)$. By the definition of conditionalization in Section 4.4, we have $P' : V' \longrightarrow [0, 1]$. Let

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

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

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

We can see that N represents the sum of the probabilities of those rows which satisfy the conditional selection condition $\bar{x} \in \{x_1, \ldots, x_h\}$. Then we know that $P'(\bar{y})$ computes the conditional probability for any $\bar{y} \in dom(V - \{v\})$.

■

Proof: Proof of Theorem 4
Let $S = \langle T, V, P, C, \omega \rangle$ and $S' = \langle T', V', P', C, \omega' \rangle$ be two Cartesian product-compatible SPOs. We prove that the Cartesian product gives the correct joint probability distribution.

Let $P(\bar{x})$ and $P'(\bar{y})$ be two probability distributions with $\bar{x} \in dom(V)$ and $\bar{y} \in dom(V')$. Since we assume that the variables in the two SPOs are independent, the definition of Cartesian product in Section 4.5, $P''(\bar{x}, \bar{y}) = P(\bar{x}) \cdot P'(\bar{y})$, correctly computes the joint probability distribution by multiplying the probabilities of the individual events.

■

Proof: Proof of Lemma 1
Let $\bar{x} \in dom(V_1)$, $\bar{y} \in dom(V_c)$, $\bar{z} \in dom(V_1')$, and V_1, V_c and V_1' be disjoint. We prove, under the assumption of independence between variables in V_1 and V_1', the following equation holds:

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

By the definition of conditional probability, we have

$$\begin{aligned} P(\bar{x}, \bar{y}, \bar{z}) &= P((\bar{x}, \bar{z}), \bar{y}) \\ &= P((\bar{x}, \bar{z})|\bar{y}) \cdot P(\bar{y}) \\ &= P(\bar{x}|\bar{y}, \bar{z}|\bar{y}) \cdot P(\bar{y}). \end{aligned}$$

The assumption that $\bar{x}$ and $\bar{z}$ are independent implies that $\bar{x}|\bar{y}$ and $\bar{z}|\bar{y}$ are independent, so

$$\begin{aligned} P(\bar{x}, \bar{y}, \bar{z}) &= P((\bar{x}|\bar{y}) \cdot P(\bar{z}|\bar{y}) \cdot P(\bar{y}) \\ &= P(\bar{x}, \bar{y}) \cdot P(\bar{y}, \bar{z})/P(\bar{y}) \\ &= P(\bar{x}, \bar{y}) \cdot P(\bar{z}|\bar{y}) \\ &= P(\bar{z}, \bar{y}) \cdot P(\bar{x}|\bar{y}). \end{aligned}$$

■

Proof: Proof of Theorem 5
Let $S = \langle T, V, P, C, \omega\rangle$ and $S' = \langle T', V', P', C, \omega'\rangle$ be two join-compatible SPOs. Here we prove that the left join operation gives the correct joint probability distribution. Note that the case of the right join is completely analogous.

Let $P(\bar{x}, \bar{y})$ and $P'(\bar{y}, \bar{z})$ be two probability distributions with $\bar{x} \in dom(V_1)$, $\bar{y} \in dom(V_c)$ and $\bar{z} \in dom(V_1')$ By assuming that the variables $\bar{x}$ and $\bar{z}$ are independent, Lemma 1 gives $P(\bar{x}, \bar{y}, \bar{z}) = P(\bar{x}, \bar{y}) \cdot P'(\bar{y}, \bar{z})/P'(\bar{y})$. So the definition of left join in Section 4.6 computes the joint probability distribution of random variables in V''.

■

Proof: Proof of Theorem 6
Let S and S' be two join-compatible SPOs. We prove that the left join and the right join are equivalent if and only if the two SPOs are *join-consistent*.

If the two SPOs are *join-consistent*, then the join operations $S \ltimes S'$ and $S \rtimes S'$ are equivalent. From the definition of *join-consistent*, we know that $P(\bar{y}) = P'(\bar{y})$ for any $\bar{y} \in dom(V_c)$ and $V_c \neq \emptyset$. Then the probability distribution $P(\bar{x}, \bar{y}, \bar{z})$ will be identical, which implies the the two join operations $S \ltimes S'$ and $S \rtimes S'$ are equivalent.

Second, consider the other direction. If the join operations $S \ltimes S'$ and $S \rtimes S'$ are equivalent, i.e. $P''(\bar{w}) = P'''(\bar{w})$ for any $\bar{w} \in dom(V'')$, then by the definition of both join operations we see that

$$P'(\bar{y}) = P(\bar{x}, \bar{y}) \cdot P'(\bar{y}, \bar{z})/P''(\bar{w}),$$

and

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

so the two probability distributions are identical, i.e. $P(\bar{y}) = P'(\bar{y})$ for any $\bar{y} \in dom(V_c)$, which implies that the two SPOs are *join-consistent*.

■