Probabilistic Object Bases

Thomas Eiter* James J. Lu† Thomas Lukasiewicz† V.S. Subrahmanian‡

Abstract

There are many applications where an object oriented data model is a good way of representing and querying data. However, current object database systems are unable to handle the case of objects whose attributes are uncertain. In this paper, extending previous pioneering work by Kornatzky and Shimony, we develop an extension of the relational algebra to the case of object bases with uncertainty. We propose concepts of consistency for such object bases, together with an NP-completeness result, and classes of probabilistic object bases for which consistency is polynomially checkable. In addition, as certain operations involve conjunctions and disjunctions of events, and as the probability of conjunctive and disjunctive events depends both on the probabilities of the primitive events involved as well as on what is known (if anything) about the relationship between the events, we show how all our algebraic operations may be performed under arbitrary probabilistic conjunction and disjunction strategies. We also develop a host of equivalence results in our algebra, which may be used as rewrite rules for query optimization. Last but not least, we have developed a prototype probabilistic object base server using the VisiBroker ORB on top of ObjectStore. We describe experiments to assess the efficiency of different possible rewrite rules.

1 Introduction

The concept of an *object base* is gaining numerous adherents because it allows data to be organized in an application specific manner for scalability, while still supporting a common query language. However, there are many applications where *probabilistic* data needs to be stored. For instance, image interpretation programs are uncertain in their identification of features in images and such image databases are typically stored using object databases [17]. Similarly, an application that is tracking a set of mobile objects using an object database system may only know that an object is at one of a given set of points right now, but the precise location may be unknown. Likewise, an application that represents forecasts about stock movements or the weather needs to represent uncertainty in the forecast. When the application data (stocks, weather) are in an object repository, methods to represent

*Technische Universität Wien, Institut für Informationssysteme, Abteilung für Wissensbasierte Systeme, Favoritenstraße 9-11, A-1040 Vienna, Austria. E-mail: eiter@kr.tuwien.ac.at, lukasiewicz@kr.tuwien.ac.at

†Department of Computer Science, Bucknell University, Lewisburg, PA 17837, USA. E-mail: jameslu@bucknell.edu

‡Institute for Advanced Computer Studies, Institute for Systems Research and Department of Computer Science, University of Maryland, College Park, Maryland 20742. E-mail: vs@cs.umd.edu.

uncertain aspects of these objects need to be developed. In short, the ability to represent probabilistic information in an object base, and to manipulate such "probabilistic object bases" efficiently is important for a variety of applications.

To date, there has been only one significant attempt in the database community to merge probability models with object bases, namely that by Kornatzky and Shimony [23], who proposed a probabilistic object calculus. Building upon their influential work, in this paper, we make the following contributions:

1. First and foremost, we propose a notion of a probabilistic schema and formally define a logical model theory for it. We define what *consistent schemas* are and prove that consistency checking is NP-complete. We identify special classes of schemas for which consistency may be polynomially checked. Previous work on probabilistic object bases had no associated concept of consistency.

2. We then propose an algebra for probabilistic object bases in which the classical relational algebra operators are extended to apply to probabilistic object bases. It is well known [24] that the probabilities of conjunctive and disjunctive events are computed in different ways depending upon the dependencies between the events involved. Our algebraic operators are parameterized by the user's knowledge (or lack thereof) of such dependencies — hence, the user can ask queries of the form "Find the join of . . . assuming no knowledge about the dependencies between the events involved." Previous work on probabilistic object bases assumed that all events involved were independent. To our knowledge, this is the first (extension of the) relational algebra for probabilistic object bases.

3. We then prove a host of equivalence results in our algebra. These equivalence results may be used as the set of rewrite rules that a database query optimizer uses for query rewriting.

4. We have implemented a distributed probabilistic object base system on top of the VisiBroker ORB and the ObjectStore commercial relational database system. This implementation allowed us to conduct experiments across the network to evaluate the performance of our system and also to see how to rewrite queries.

This paper is structured as follows. In the next section, we consider a database application which motivates our approach. In Section 3, we describe the architecture of a probabilistic object base system. After some basic definitions of probability concepts in Section 4, we develop our probabilistic object base (POB) model in Sections 5 and 6. A POB-algebra for querying this model is then presented in Section 7, and equivalence results in this algebra are derived in Section 8. We report on an implementation of the model in Section 9, and discuss related work in Section 10. Section 11 concludes the paper.

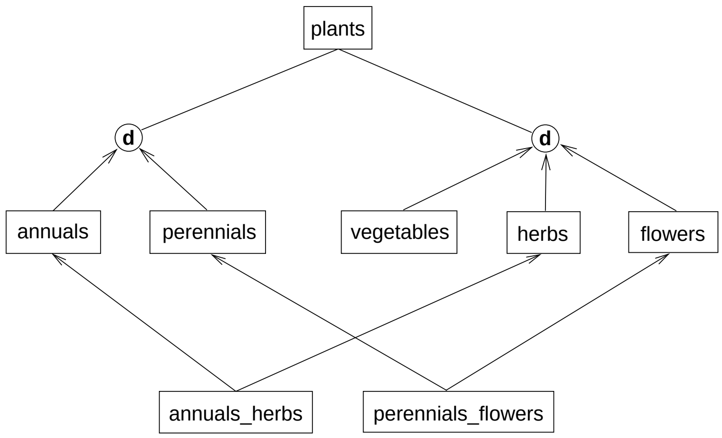


Figure 1: Plant identification example

2 A Motivating Example

Consider the task of building an extensive database describing the types of vegetation found in the Amazon rainforest. The creation of such a database is a formidable task. Individuals need to exhaustively examine the vegetables, herbs, and other kinds of plants growing in these forests, and provide information describing soil conditions, climactic conditions, etc.

When describing the plants growing in such rainforests, there are several possible causes of *uncertainty*. First and foremost, some plant species may not be uniquely identifiable by the surveyor in the field. He may classify a particular herb as either being Silver Thyme or French Thyme (two different species of thyme), without being able to specify exactly which specie the plant in question belongs to. By the same token, if he were slightly more expert, he might be able to say that he is not sure whether the herb is Silver Thyme or French Thyme, but he rates the probability that it is Silver Thyme twice as high as that it is French Thyme.

Figure 1 shows a very simple class hierarchy that describes plants as either being perennials or annuals, and either being vegetables, herbs, or flowers. Clearly, the classes perennials and annuals are disjoint (that is, a plant cannot be both an annual and a perennial), as are the classes vegetables, herbs, and flowers. The classes that are mutually disjoint are connected together by a "**d**" in Figure 1. However, note that we can certainly have plants that are annuals and herbs (for example, Basil).

In the rest of this paper, we will repeatedly consider this example, in order to illustrate various definitions. By the time this paper is completed, we would have described techniques to build and query a probabilistic object base that captures the Plant Database of this example as a special case.

3 Architecture of a Probabilistic Object Base

In this section, we describe the overall architecture of a POB system. Figure 2 presents an architecture for query processing in probabilistic object bases. The architecture consists of the following components:

- The user expresses queries through a graphical user interface. As a result of the user's interaction with the interface, the GUI generates as output, a query in a declarative *probabilistic object calculus* (POC). Note that queries in this calculus are declarative queries. A pioneering attempt at such a calculus is that of Kornatzky and Shimony [23].

- The calculus query generated will be fed into a *Converter* which converts probabilistic object calculus queries into queries in a probabilistic object algebra query.

- The algebraic query generated by the converter will be fed into a *Query Optimizer*, which will take as input a set of *rewrite rules* (reflecting equivalences between different queries in the POA-algebra) and a set of *cost models* to perform the optimization. Note that given a set of rewrite rules and a set of cost models, the task of finding a rewriting of a query that has minimal expected cost (according to the cost models) is well-studied, and good commercial implementations of such code exist (e.g. Grafe's CASCADES system is presently being used by Microsoft).

- The "optimized" algebra query then produced will be physically executed on the probabilistic object base.

- All the components above will use *libraries* consisting of: (i) A set of probabilistic *conjunction, disjunction and difference* strategies that allow the user to express what she knows about the dependencies between events she is querying about — this is used in query formulation, query optimization, cost evaluation and query execution. (ii) A set of distribution functions that allow a user to specify how probabilities are distributed over a space of possible values for an unknown attribute.

Giving a detailed description of all these components is clearly beyond the scope of a single paper. In previous work, Kornatzky and Shimony [23] developed a probabilistic object calculus. In this paper, we will expand the concept of a probabilistic object base used by them. We will then define formally an Probabilistic Object Algebra (POA) and prove a host of query equivalence results. We will report on a prototype implementation of the POA and describe experimental results — given a query equivalence $q_1 = q_2$, these experimental results will identify when a query of the form q_1 should be rewritten to a query of the form q_2 and vice versa. To our knowledge, this paper is the first to propose a probabilistic object algebra, the first to present results on query equivalences in such an algebra, and the first to implement such an algebra on top of a commercial object database system.

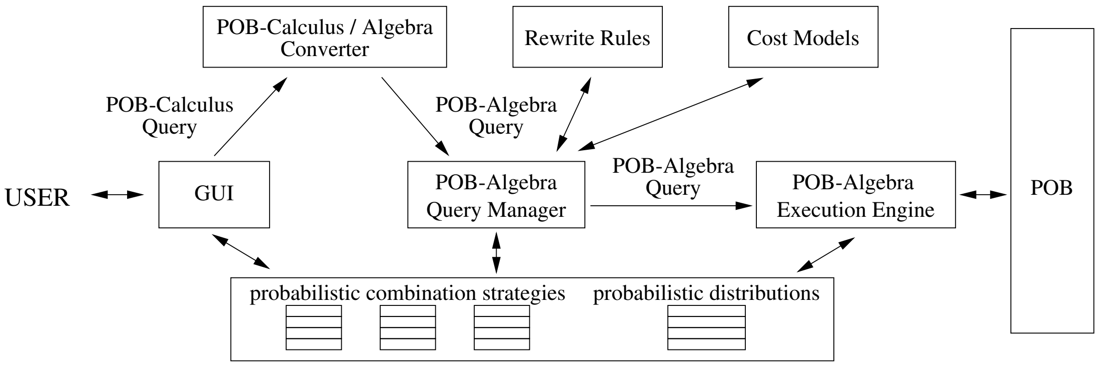


Figure 2: Architecture of POB system

4 Basic Probability Definitions

In this section, we present some basic definitions used to set up a probabilistic extension of object bases. The probabilistic concepts are divided into two parts — (i) probabilistic conjunction, disjunction, and difference strategies, and (ii) distribution functions.

4.1 Probabilistic combination strategies

Suppose we know the probabilities of some events e_1 and e_2. For example, e_1 may be the event "The photographed plant p_1 (in image I) is French Thyme." Similarly, e_2 may be the event "The photographed plant p_2 (in image I) is Mint." Assume now that we are interested in the probability of the complex event $(e_1 \wedge e_2)$. The probability of $(e_1 \wedge e_2)$ is computed in different ways based upon our knowledge of the dependencies between e_1 and e_2.

- *e_1 and e_2 are independent.* This may occur if *we know* that the plants p_1 and p_2 are growing in the area independently of each other. In this case, $\mathbf{P}(e_1 \wedge e_2) = \mathbf{P}(e_1) \cdot \mathbf{P}(e_2)$ (that is, the probability of $(e_1 \wedge e_2)$ is the product of the probabilities of e_1 and e_2).

- *e_1 and e_2 are mutually exclusive.* Suppose, for example, we know that the plants p_1 and p_2 are both either Thyme or Mint (e.g., if, for some reason, they cannot grow together in the same place). This means that the events e_1 and e_2 are mutually exclusive, and so we can immediately say that $\mathbf{P}(e_1 \wedge e_2) = 0$.

- *We are ignorant of the relationship between e_1 and e_2.* This case occurs when we do not know anything about the relationship between the plants p_1 and p_2 growing in the same area.

  In this case, as already shown by Boole [5], the best we can say about the probability of $(e_1 \wedge e_2)$ is that it lies in the interval $[\max(0, \mathbf{P}(e_1) + \mathbf{P}(e_2) - 1), \min(\mathbf{P}(e_1), \mathbf{P}(e_2))]$.

Thus, the probability of $(e_1 \wedge e_2)$ depends not only on the probabilities of e_1 and e_2, but also on the relationship between events e_1 and e_2. A similar situation applies when we consider complex events such as $(e_1 \vee e_2)$. The above are only three *examples* of different ways of evaluating probabilities of complex events. In general, depending on exactly what is known about the dependencies between the events involved, there is a whole plethora of such probability computations.

In our framework, we use probability intervals instead of point probabilities for two reasons: (i) In many applications, the probability of an event is often not precisely given; (ii) as shown by Boole [5] back in 1854, when we do not know the dependencies between two events, all we can say is that the probability of the conjunction/disjunction of the events can only be specified as an interval.

Definition 4.1 (consistent assignment of probabilistic intervals to two events) Suppose e_1 and e_2 have probabilities in the intervals $I_1 = [L_1, U_1]$ and $I_2 = [L_2, U_2]$, respectively. Such an assignment of probabilistic intervals is *consistent* iff the following conditions hold:

- If $(e_1 \wedge e_2)$ is contradictory[1] , then $L_1 + L_2 \leq 1$.
- If $(e_1 \wedge \neg e_2)$ is contradictory, then $L_1 \leq U_2$.
- If $(\neg e_1 \wedge e_2)$ is contradictory, then $L_2 \leq U_1$.
- If $(\neg e_1 \wedge \neg e_2)$ is contradictory, then $U_1 + U_2 \geq 1$.

In the sequel, all assignments of probabilistic intervals are implicitly assumed to be consistent unless stated otherwise. We denote for intervals $I_1 = [L_1, U_1]$ and $I_2 = [L_2, U_2]$ by $I_1 \leq I_2$ that $L_1 \leq L_2$ and $U_1 \leq U_2$, and by $I_1 \subseteq I_2$ that I_1 is contained in I_2, i.e., $L_2 \leq L_1$ and $U_1 \leq U_2$.

As many relationships between events cannot be automatically inferred, it is imperative that the user be able to specify, in his query, what knowledge he has about such relationships. To facilitate this, Lakshmanan *et al.* [24] have introduced generic conjunction and disjunction strategies. Any function that satisfies the axioms listed below is called a conjunction or disjunction strategy, respectively. (Given two events e_1 and e_2 with probabilities in the intervals $I_1 = [L_1, U_1]$ and $I_2 = [L_2, U_2]$, respectively, the notations "$I = I_1 \otimes I_2$" and "$I = I_1 \oplus I_2$" are shorthand for "$(e_1 \wedge e_2, I) = (e_1, I_1) \otimes (e_2, I_2)$" and "$(e_1 \vee e_2, I) = (e_1, I_1) \oplus (e_2, I_2)$", respectively.)

Axiom Name	Conjunction Strategy	Disjunction Strategy
Bottomline	$(I_1 \otimes I_2) \leq [\min(L_1, L_2), \min(U_1, U_2)]$	$(I_1 \oplus I_2) \geq [\max(L_1, L_2), \max(U_1, U_2)]$
Ignorance	$(I_1 \otimes I_2) \subseteq [\max(0, L_1 + L_2 - 1), \min(U_1, U_2)]$	$(I_1 \oplus I_2) \subseteq [\max(L_1, L_2), \min(1, U_1 + U_2)]$
Identity*	$(I_1 \otimes [1, 1]) = I_1$	$(I_1 \oplus [0, 0]) = I_1$
Commutativity	$(I_1 \otimes I_2) = (I_2 \otimes I_1)$	$(I_1 \oplus I_2) = (I_2 \oplus I_1)$
Associativity	$((I_1 \otimes I_2) \otimes I_3) = (I_1 \otimes (I_2 \otimes I_3))$	$((I_1 \oplus I_2) \oplus I_3) = (I_1 \oplus (I_2 \oplus I_3))$
Monotonicity	$(I_1 \otimes I_2) \leq (I_2 \otimes I_3)$ if $I_2 \leq I_3$	$(I_1 \oplus I_2) \leq (I_1 \oplus I_3)$ if $I_2 \leq I_3$

[1] Contradictory here merely means "inconsistent in classical propositional logic."

In the above table, the Identity-axioms for conjunction and disjunction strategies assume that $e_1 \wedge e_2$ and $\neg e_1 \wedge \neg e_2$, respectively, are not contradictory.

While the notion of probabilistic conjunction and disjunction strategies are recapitulated from Lakshmanan *et al.* [24], the concept of difference strategies below is new.

Definition 4.2 (probabilistic difference strategy) Suppose e_1 and e_2 have probabilities in the intervals $I_1 = [L_1, U_1]$ and $I_2 = [L_2, U_2]$, respectively. A *probabilistic difference strategy* is a binary operation $\ominus$ that uses this information to compute the probabilistic interval $I = [L, U]$ for the event $(e_1 \wedge \neg e_2)$. When the events involved are clear from context, we use "$I = I_1 \ominus I_2$" to denote "$(e_1 \wedge \neg e_2, I) = (e_1, I_1) \ominus (e_2, I_2)$". Every difference strategy must conform to the following postulates of probabilistic difference:

1. Bottomline: $(I_1 \ominus I_2) \leq [\min(L_1, 1 - U_2), \min(U_1, 1 - L_2)]$.
2. Ignorance: $(I_1 \ominus I_2) \subseteq [\max(0, L_1 - U_2), \min(U_1, 1 - L_2)]$.
3. Identity: If $(\neg e_1 \wedge \neg e_2)$ is not contradictory, then $(I_1 \ominus [0, 0]) = I_1$.

Some examples of probabilistic conjunction, disjunction, and difference strategies are given in Table 1.

Strategy	Operators
Ignorance	$([L_1, U_1] \otimes_{ig} [L_2, U_2]) \equiv [\max(0, L_1 + L_2 - 1), \min(U_1, U_2)]$
	$([L_1, U_1] \oplus_{ig} [L_2, U_2]) \equiv [\max(L_1, L_2), \min(1, U_1 + U_2)]$
	$([L_1, U_1] \ominus_{ig} [L_2, U_2]) \equiv [\max(0, L_1 - U_2), \min(U_1, 1 - L_2)]$
Independence	$([L_1, U_1] \otimes_{in} [L_2, U_2]) \equiv [L_1 \cdot L_2, U_1 \cdot U_2]$
	$([L_1, U_1] \oplus_{in} [L_2, U_2]) \equiv [L_1 + L_2 - (L_1 \cdot L_2), U_1 + U_2 - (U_1 \cdot U_2)]$
	$([L_1, U_1] \ominus_{in} [L_2, U_2]) \equiv [L_1 \cdot (1 - U_2), U_1 \cdot (1 - L_2)]$
Positive Correlation	$([L_1, U_1] \otimes_{pc} [L_2, U_2]) \equiv [\min(L_1, L_2), \min(U_1, U_2)]$
(when e_1 implies e_2,	$([L_1, U_1] \oplus_{pc} [L_2, U_2]) \equiv [\max(L_1, L_2), \max(U_1, U_2)]$
or e_2 implies e_1)	$([L_1, U_1] \ominus_{pc} [L_2, U_2]) \equiv [\max(0, L_1 - U_2), \max(0, U_1 - L_2)]$
Negative Correlation	$([L_1, U_1] \otimes_{nc} [L_2, U_2]) \equiv [0, 0]$
(when e_1 and e_2 are	$([L_1, U_1] \oplus_{nc} [L_2, U_2]) \equiv [\min(1, L_1 + L_2), \min(1, U_1 + U_2)]$
mutually exclusive)	$([L_1, U_1] \ominus_{nc} [L_2, U_2]) \equiv [L_1, \min(U_1, 1 - L_2)]$

Table 1: Examples of probabilistic combination strategies

Note that we do not assume any postulates that relate probabilistic conjunction, disjunction, and difference strategies to each other (for example, postulates that express the distributivity of conjunction and disjunction strategies). Readers may make such assumptions if they wish — however, the results of this paper stand even if these assumptions are not made.

4.2 Probability distribution functions

Probability distribution functions assign probabilities to elementary events in a coherent way. For example, if we are told that plant p_1 is currently at one of the locations a, b, c with probability 60-70%, then a distribution function allows us to assign parts of this probability mass to the events "plant p_1 is at location a," "plant p_1 is at location b," and "plant p_1 is at location c."

Definition 4.3 (distribution function) Let X be a finite set. A *(probability) distribution function* α *over* X is a mapping from X to the real interval $[0, 1]$ such that $\Sigma_{x \in X}\, \alpha(x) \leq 1$.

We do not require that $\Sigma_{x \in X}\, \alpha(x) = 1$ holds; we call any distribution function α with this property *complete*. The above definition allows to conveniently assign probabilities to a subset $X \subseteq Y$ of elements, leaving the probabilities of the other elements open.

An important distribution function, which we often encounter in practice, is the *uniform distribution*. For a finite set X, it is defined by $\mathrm{u}_X(x) = \frac{1}{|X|}$ for all $x \in X$. We abbreviate u_X by u, whenever X is clear from the context. Many other distribution functions are conceivable; we will not embark on a study of this subject here.

Definition 4.4 (probabilistic triple) A *probabilistic triple* $\langle X, \alpha, \beta \rangle$ consists of a finite set X, a distribution function α over X, and a function $\beta : X \to [0, 1]$ mapping X to the real interval $[0, 1]$ such that (i) $\alpha(x) \leq \beta(x)$ for all $x \in X$ and (ii) $\sum_{x \in X} \beta(x) \geq 1$ hold.

Informally, a probabilistic triple assigns each element x of a set X a probability interval $[\alpha(x), \beta(x)]$. This assignment is consistent in the sense that we can assign each element in X a probability $p(x)$ from $[\alpha(x), \beta(x)]$ such that the sum of all $p(x)$ adds up to 1.

5 Types and Probabilistic Object Base Schemas

In this section, we provide some basic definitions underlying a probabilistic object base (POB). We first consider types and values, and then the schema of a POB. The notion of POB-schema is more complex than in the context of relational databases, and may lead to inconsistent specifications; we present efficient algorithms for checking schema consistency.

5.1 Types and values

We start with the definition of types.

Definition 5.1 (types) Let $\mathcal{A}$ be a set of *attributes* and let $\mathcal{T}$ be a set of *atomic types*. We define *types* inductively as follows:

- Every atomic type from $\mathcal{T}$ is a type.

- If τ is a type, then $\{\tau\}$ is a type, which is called the set type of τ;

- If $A_1, \ldots, A_k$ are pairwise different attributes from $\mathcal{A}$ and $\tau_1, \ldots, \tau_k$ are types, then $[A_1\colon \tau_1, \ldots, A_k\colon \tau_k]$ is a type. This type is called a *tuple type* over the set of attributes $\{A_1, \ldots, A_k\}$. Given such a type $\tau = [A_1\colon \tau_1, \ldots, A_k\colon \tau_k]$, we use $\tau.A_i$ to denote τ_i.

Example 5.1 (Plant Example: types) In the Plant Example, some atomic types from $\mathcal{T}$ may be given by $\mathsf{integer}$, real, string, $\mathsf{soiltype}$, and $\mathsf{suntype}$. The attributes soil, sun (sun-exposure), and rain (daily water) describe various conditions needed for a plant to grow. Some (non atomic) types include: $\mathsf{soiltype}$; $\{\mathsf{soiltype}\}$; and $[\mathsf{soil}\colon \{\mathsf{soiltype}\}, \mathsf{sun}\colon \mathsf{suntype}, \mathsf{rain}\colon \mathsf{integer}]$.

Definition 5.2 (values) Every atomic type $\tau \in \mathcal{T}$ has an associated *domain* $\mathrm{dom}(\tau)$. We define *values* by induction as follows:

- For all atomic types $\tau \in \mathcal{T}$, every $v \in \mathrm{dom}(\tau)$ is a value of type τ.

- If $v_1, \ldots, v_k$ are values of type τ, then $\{v_1, \ldots, v_k\}$ is a value of type $\{\tau\}$.

- If $A_1, \ldots, A_k$ are pairwise different attributes from $\mathcal{A}$ and $v_1, \ldots, v_k$ are values of types $\tau_1, \ldots, \tau_k$, then $[A_1\colon v_1, \ldots, A_k\colon v_k]$ is a value of type $[A_1\colon \tau_1, \ldots, A_k\colon \tau_k]$.

Example 5.2 (Plant Example: values) Let us return to the types of Example 5.1. We assign the usual domains to $\mathsf{integer}$, real, and string. Let $\mathsf{soiltype}$ and $\mathsf{suntype}$ be enumerated types having the domains $\{\mathsf{loamy}, \mathsf{swampy}, \mathsf{sandy}\}$ and $\{\mathsf{mild}, \mathsf{medium}, \mathsf{heavy}\}$, respectively. The value sets associated with the types of Example 5.1 are as follows:

- $\mathsf{soiltype}$: Any element of $\{\mathsf{loamy}, \mathsf{swampy}, \mathsf{sandy}\}$ is a value of $\mathsf{soiltype}$. For example, loamy is a value of $\mathsf{soiltype}$. When associated with a particular plant, this value might say that the plant needs loamy soil to flourish.

- $\{\mathsf{soiltype}\}$: Any set of values of $\mathsf{soiltype}$ is a value of this type. For example, if a particular plant can grow well in either loamy or swampy soil, then $\{\mathsf{loamy}, \mathsf{swampy}\}$ is an appropriate value of this type that can be associated with this plant.

- $[\mathsf{soil}\colon \{\mathsf{soiltype}\}, \mathsf{sun}\colon \mathsf{suntype}, \mathsf{rain}\colon \mathsf{integer}]$: Any triple (v_1, v_2, v_3) is a value of this type, where v_1 is a *set* of values of $\mathsf{soiltype}$, v_2 is a value of $\mathsf{suntype}$, and v_3 is a value of $\mathsf{integer}$. For example, $(\{\mathsf{loamy}, \mathsf{swampy}\}, \mathsf{mild}, 3)$ is a value of this type. It says that the plant needs either loamy or swampy soil, mild sun, and 3 units of water per day to flourish.

Definition 5.3 (probabilistic tuple values) If $A_1, \ldots, A_k$ are pairwise different attributes from $\mathcal{A}$ and $(V_1, \alpha_1, \beta_1), \ldots, (V_k, \alpha_k, \beta_k)$ are probabilistic triples where $V_1, \ldots, V_k$ are *sets* of values of types $\tau_1, \ldots, \tau_k$, then the expression $[A_1\colon (V_1, \alpha_1, \beta_1), \ldots, A_k\colon (V_k, \alpha_k, \beta_k)]$ is a probabilistic tuple value of type $[A_1\colon \tau_1, \ldots, A_k\colon \tau_k]$ over the set of attributes $\{A_1, \ldots, A_k\}$. For probabilistic tuple values $ptv = [A_1\colon (V_1, \alpha_1, \beta_1), \ldots, A_k\colon (V_k, \alpha_k, \beta_k)]$, we use $ptv.A_i$ to denote (V_i, α_i, β_i).

It is important to note that the order of the $A_i\colon (V_i, \alpha_i, \beta_i)$'s in a probabilistic tuple value $ptv = [A_1\colon (V_1, \alpha_1, \beta_1), \ldots, A_k\colon (V_k, \alpha_k, \beta_k)]$ is not important. That is, we adopt a set-oriented view of probabilistic tuple values.

Example 5.3 (Plant Example: probabilistic tuple values) Let us consider a specific plant growing wild in a forest. We know that the soil type of this plant is loamy (presumably, as we can see, the plant is flourishing in the place in which it is currently growing in). Moreover, we are sure that this plant is Thyme, but unsure whether it is French Thyme (french), Silver Thyme (silver) or Wooly Thyme (wooly). If we are sure with 20–60% probability each that it is French Thyme, Silver Thyme, and Wooly Thyme, then we may encode this knowledge via the following probabilistic tuple value of type $[\mathsf{soil}\colon \mathsf{soiltype},\ \mathsf{classification}\colon \mathsf{string}]$ over the set of attributes $\{\mathsf{soil}, \mathsf{classification}\}$:

$$[\mathsf{soil}\colon \langle\{\mathsf{loamy}\}, \mathrm{u}, \mathrm{u}\rangle,\ \mathsf{classification}\colon \langle\{\mathsf{french}, \mathsf{silver}, \mathsf{wooly}\}, 0.6\,\mathrm{u}, 1.8\,\mathrm{u}\rangle].$$

Note that the expressions "$0.6\,\mathrm{u}$" and "$1.8\,\mathrm{u}$" denote the distribution function α and the function β, respectively, that are defined by $\alpha(x) = 0.6{\cdot}1/3$ and $\beta(x) = 1.8{\cdot}1/3$ for all x from $\{\mathsf{french}, \mathsf{silver}, \mathsf{wooly}\}$.

In the above definition, a probabilistic triple (V_i, α_i, β_i) may only assign a probability interval to some values v (viz. those in V_i) for the attribute A_i. Nothing is stated for the (possibly infinitely many) other values that A_i could have according to its type τ_i. We must find a clean and appealing way in which such incomplete knowledge about the probability assignment is handled.

In the tradition of relational databases, we adopt a form of the *closed world assumption* (CWA): We assume that every value $v \in \mathrm{dom}(\tau_i) - V_i$ has probability 0, i.e., it is implicitly assigned the probability interval $[0, 0]$. Under this convention, "consistency" (which we will define formally later) of the probability information given by (V_i, α_i, β_i) is preserved in the larger context of $\mathrm{dom}(\tau_i)$: a probability function p over $\mathrm{dom}(\tau_i)$ exists, compatible with (V_i, α_i, β_i), such that the sum of all $p(v)$, $v \in \mathrm{dom}(\tau_i)$, is 1.

This CWA will underly our definition of the operations in the probabilistic object base algebra in Section 7. Notice that still an open world view is possible for particular values. We may, for instance, add v to V and set $\alpha(v) = 0, \beta(v) = 1$; this explicitly expresses that the probability of v is unknown.

5.2 Probabilistic object base schema

Informally, a probabilistic object base schema consists of a hierarchy of classes. Membership of an object in an immediate subclass of any class is expressed by a probability value.

Definition 5.4 (probabilistic object base schema) A *probabilistic object base schema (POB-schema)* is a quintuple $(\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$, where:

- $\mathcal{C}$ *is a finite set of* classes. Intuitively, these reflect the classes associated with this probabilistic object base.

- *σ maps each class from $\mathcal{C}$ to a tuple type.* Intuitively, this mapping specifies the data type of each class.

- *$\Rightarrow$ is a binary relation on $\mathcal{C}$ such that $(\mathcal{C}, \Rightarrow)$ is a directed acyclic graph (dag).* Intuitively, each node of the directed acyclic graph $(\mathcal{C}, \Rightarrow)$ is a class from $\mathcal{C}$ and each edge $c_1 \Rightarrow c_2$ says that the class c_1 is an *immediate subclass* of c_2.

- me *maps each class c to a partition of the set of all immediate subclasses of c.* Intuitively, suppose that the class c has the five subclasses $c_1, \ldots, c_5$ and suppose that $\mathrm{me}(c)$ is the partition $\{\{c_1, c_2\}, \{c_3, c_4, c_5\}\}$. Here, $\mathrm{me}(c)$ produces two clusters. An object o belonging to class c can belong to either or both clusters of c. However, the classes within a cluster are mutually exclusive, that is, o cannot belong to both c_1 and c_2 in the same time.

- *$\wp$ maps each edge in $(\mathcal{C}, \Rightarrow)$ to a positive rational number in the unit interval $[0, 1]$ such that for all classes c and all clusters $\mathcal{P} \in \mathrm{me}(c)$, it holds that $\Sigma_{d \in \mathcal{P}} \wp(d, c) \leq 1$.* Intuitively, if $c_1 \Rightarrow c_2$, then $\wp(c_1, c_2)$ specifies the conditional probability that an arbitrary object belongs to the subclass c_1 given that it belongs to the superclass c_2. The summation condition says that the sum of the probabilities of edges within a mutually exclusive set of subclasses must sum up to less than or equal to 1.

A *directed path* in the directed acyclic graph $(\mathcal{C}, \Rightarrow)$ is a sequence of classes $c_1, c_2, \ldots, c_k$ such that $c_1 \Rightarrow c_2 \Rightarrow \cdots \Rightarrow c_k$ and $k \geq 1$. We use $\Rightarrow^\star$ to denote the reflexive and transitive closure of $\Rightarrow$. Note that $\Rightarrow^\star$ induces a natural partial order $\leq$ on $\mathcal{C}$ by $c \leq d$ iff $c \Rightarrow^\star d$ for all $c, d \in \mathcal{C}$.

We use $\mathrm{S}(c) = \{d \in \mathcal{C} \mid d \Rightarrow c\}$ to denote the set of all *immediate subclasses* of $c \in \mathcal{C}$, and $\mathrm{S}^\star(c) = \{d \in \mathcal{C} \mid d \Rightarrow^\star c\}$ to denote the set of *subclasses* of $c \in \mathcal{C}$. A class d is a *subclass* of a partition cluster $\mathcal{P}$ iff d is a subclass of some $c \in \mathcal{P}$.

We will represent the above structure (excluding the type assignment σ) in a graphical way as shown in Figure 3, where the edges are labeled by conditional probabilities.

Example 5.4 (Plant Example: probabilistic object base schema) A POB-schema for the Plant Example may consist of the following components:

- $\mathcal{C} = \{\mathsf{plants}, \mathsf{annuals}, \mathsf{perennials}, \mathsf{vegetables}, \mathsf{herbs}, \mathsf{flowers}, \mathsf{annuals\_herbs}, \mathsf{perennials\_flowers}\}$.

- σ is given by Table 2.

- $(\mathcal{C}, \Rightarrow)$ is the graph resulting from Figure 1, if the d-nodes are contracted to plants.

- me is the partitioning of edges shown in Figure 1.

- $\wp$ is the probability assignment in Table 2.

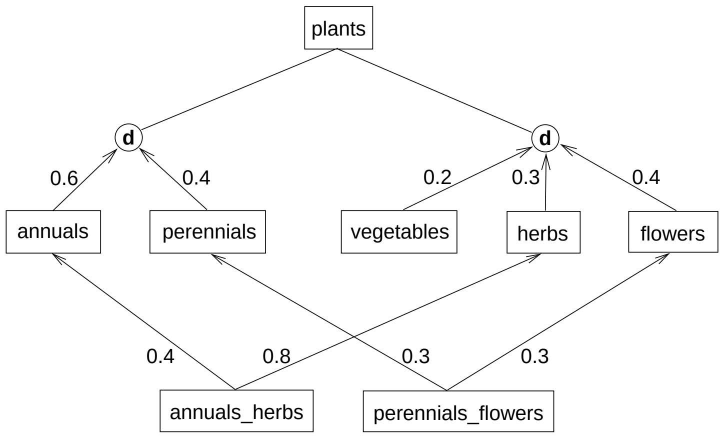


Figure 3: Plant Example with probability assignment

For example, annuals and annual_herbs are subclasses of plants, and annuals is an immediate subclass of plants while annual_herbs is not; annual_herbs is a subclass of the cluster $\{$annuals, perennials$\}$.

The POB-schemas defined so far may be inconsistent. That is, it may not always be possible to find a set of objects that satisfies the taxonomic and probabilistic knowledge expressed by the directed acyclic graph, the partitioning of edges, and the probability assignment.

More formally, the consistency of a POB-schema is defined as follows.

Definition 5.5 (consistent POB-schema) Let $\mathrm{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ be a POB-schema. An *interpretation* of S is any mapping ε from $\mathcal{C}$ to the set of all finite subsets of a set $\mathcal{O}$. An interpretation ε of S is called a *taxonomic model* of S iff it satisfies the following conditions:

C1 $\varepsilon(c) \neq \emptyset$, for all classes $c \in \mathcal{C}$.

C2 $\varepsilon(c) \subseteq \varepsilon(d)$, for all classes $c, d \in \mathcal{C}$ with $c \Rightarrow d$.

C3 $\varepsilon(c) \cap \varepsilon(d) = \emptyset$, for all distinct classes $c, d \in \mathcal{C}$ that belong to the same cluster $\mathcal{P} \in \bigcup \mathrm{me}(\mathcal{C})$.

We say that two classes $c, d \in \mathcal{C}$ are *taxonomically disjoint* (*t-disjoint*) iff $\varepsilon(c) \cap \varepsilon(d) = \emptyset$ for all taxonomic models ε of S. An interpretation ε of S is a *taxonomic and probabilistic model* (or simply *model*) of S iff it is a taxonomic model of S and it satisfies the following condition:

C4 $|\varepsilon(c)| = \wp(c, d) \cdot |\varepsilon(d)|$ for all classes $c, d \in \mathcal{C}$ with $c \Rightarrow d$.

We say S is *consistent* iff a model of S exists.

Let us illustrate this definition within the Plant Example.

Example 5.5 (Plant Example: consistent POB-schema) Let $\mathrm{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ be the POB-schema given in Example 5.4. Let $\mathcal{O}$ be a set of cardinality 800, which is partitioned into pairwise

c	$\sigma(c)$
plants	[pname : string, soil : soiltype, rain : integer]
annuals	[pname : string, soil : soiltype, rain : integer, sun : suntype]
perennials	[pname : string, soil : soiltype, rain : integer, sun : suntype, expyears : integer]
vegetables	[pname : string, soil : soiltype, rain : integer, sun : suntype, expyears : integer]
herbs	[pname : string, soil : soiltype, rain : integer, sun : suntype, expyears : integer, classification : string]
flowers	[pname : string, soil : soiltype, rain : integer, sun : suntype, expyears : integer, classification : string]
annuals_herbs	[pname : string, soil : soiltype, rain : integer, sun : suntype, expyears : integer, classification : string]
perennials_flowers	[pname : string, soil : soiltype, rain : integer, sun : suntype, expyears : integer, classification : string]

edge	probability
annuals $\Rightarrow$ plants	0.6
perennials $\Rightarrow$ plants	0.4
vegetables $\Rightarrow$ plants	0.2
herbs $\Rightarrow$ plants	0.3
flowers $\Rightarrow$ plants	0.4
annuals_herbs $\Rightarrow$ annuals	0.4
annuals_herbs $\Rightarrow$ herbs	0.8
perennials_flowers $\Rightarrow$ perennials	0.3
perennials_flowers $\Rightarrow$ flowers	0.3

Table 2: Type assignment σ and probability assignment $\wp$

disjoint subsets $\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_{10}$ having cardinalities 90, 27, 126, 45, 192, 21, 98, 35, 70, and 96, respectively. An interpretation ε of S is given by the following table:

c	$\varepsilon(c)$	$\lvert\varepsilon(c)\rvert$
plants	$\mathcal{O}_1 \cup \cdots \cup \mathcal{O}_{10}$	800
annuals	$\mathcal{O}_1 \cup \cdots \cup \mathcal{O}_5$	480
perennials	$\mathcal{O}_6 \cup \cdots \cup \mathcal{O}_{10}$	320
vegetables	$\mathcal{O}_1 \cup \mathcal{O}_9$	160
herbs	$\mathcal{O}_2 \cup \mathcal{O}_5 \cup \mathcal{O}_6$	240
flowers	$\mathcal{O}_3 \cup \mathcal{O}_7 \cup \mathcal{O}_{10}$	320
annuals_herbs	$\mathcal{O}_5$	192
perennials_flowers	$\mathcal{O}_{10}$	96

It is easy to see that ε is also a model of S. For example, $\varepsilon(\mathsf{plants}) \neq \emptyset$, $\varepsilon(\mathsf{annuals}) \subseteq \varepsilon(\mathsf{plants})$, $\varepsilon(\mathsf{annuals}) \cap \varepsilon(\mathsf{perennials}) = \emptyset$, and $|\varepsilon(\mathsf{annuals})| = 0.6 \cdot |\varepsilon(\mathsf{plants})|$. Hence, S is consistent.

It would now be nice to have an efficient algorithm for deciding the consistency of a given POB-schema. For this purpose, we need a suitable characterization of consistency. The following condition is a natural candidate.

Definition 5.6 (pseudo-consistent POB-schema) The POB-schema $\mathrm{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ is *pseudo-consistent* iff the following conditions hold:

P1 For any two different classes $c_1, c_2 \in \mathcal{C}$ with $c_1 \Rightarrow^\star c_2$, the product of the edge probabilities is the same on all paths from c_1 up to c_2.

P2 For all clusters $\mathcal{P} \in \bigcup \mathrm{me}(\mathcal{C})$, no two distinct classes $c_1, c_2 \in \mathcal{P}$ have a common subclass.

Example 5.6 (Plant Example: pseudo-consistent POB-schema) It is easy to see that the POB-schema $\mathrm{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ shown in Example 5.4 is pseudo-consistent:

- The two paths from annuals_herbs up to plants and from perennials_flowers up to plants have both 0.24 and 0.12, respectively, as the product of the edge probabilities.
- Neither annuals_herbs nor perennials_flowers is a subclass of two t-disjoint classes.

Indeed, pseudo-consistency is a necessary condition for consistency.

Theorem 5.1 *Every consistent POB-schema is pseudo-consistent.*

However, pseudo-consistency is not a sufficient condition for consistency. Even worse than that, deciding the consistency of a pseudo-consistent POB-schema is presumably intractable. We have the following result.

Theorem 5.2 *Deciding whether a given POB-schema* S *is consistent is NP-complete. Hardness holds even if* S *is pseudo-consistent.*

Proof. We show only membership in NP (the complete proof is given in [11]).

The problem is in NP, since it polynomially reduces to the NP-complete problem of deciding whether a weight formula is satisfiable in a measurable probability structure [13]. More precisely, *weight formulas* are defined as Boolean combinations of *basic weight formulas*, which are expressions of the form $a_1 \cdot w(\phi_1) + \cdots + a_k \cdot w(\phi_k) \geq a$ with integers $a_1, \ldots, a_k, a$ and propositional formulas $\phi_1, \ldots, \phi_k$. A *measurable probability structure* can be identified with a probability function on the finite set of all truth assignments to the primitive propositions, which is extended in a natural way to propositional formulas, basic weight formulas, and weight formulas.

It can now easily be shown that a POB-schema $\mathrm{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ is consistent iff the conjunction of the following weight formulas, which capture C1–C4 in Def. 5.5, is satisfiable:

C1 $\neg((-1) \cdot w(c) \geq 0)$ for all classes $c \in \mathcal{C}$.

C2 $(w(c \wedge \neg d) \geq 0) \wedge ((-1) \cdot w(c \wedge \neg d) \geq 0)$ for all classes $c, d \in \mathcal{C}$ with $c \Rightarrow d$.

C3 $(w(c \wedge d) \geq 0) \wedge ((-1) \cdot w(c \wedge d) \geq 0)$ for all distinct classes $c, d \in \mathcal{C}$ of the same cluster.

C4 $(n \cdot w(c) + (-m) \cdot w(d) \geq 0) \wedge ((-n) \cdot w(c) + m \cdot w(d) \geq 0)$ for all classes $c, d \in \mathcal{C}$ with $c \Rightarrow d$, where m and n are natural numbers such that $\wp(c, d) = \frac{m}{n}$. $\Box$

Nonetheless, polynomial algorithms for deciding the consistency of a POB-schema in relevant special cases may be possible. Well-structured POB-schemas, which we introduce next, enjoy this property.

Definition 5.7 (well-structured POB-schema) The POB-schema $\mathrm{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ is *well-structured* iff the following conditions hold:

W1 There exists a class $c \in \mathcal{C}$ such that every class $d \in \mathcal{C}$ is a subclass of c (i.e., the graph $(\mathcal{C}, \Rightarrow)$ has a *top element*).

W2 For all classes $c \in \mathcal{C}$ and every distinct $c_1, c_2 \in \mathrm{S}(c)$, the set $S := \mathrm{S}^\star(c_1) \cap \mathrm{S}^\star(c_2)$ is either empty or has a unique element $d_m \neq c_1, c_2$ such that $d \Rightarrow^\star d_m$ for all $d \in S$ (i.e., c_1,c_2 either have no common subclass or a greatest common subclass d_m, which is different from them).

W3 For every class $c \in \mathcal{C}$, the undirected graph $G_{\mathbf{S}}(c) = (\mathcal{V}, \mathcal{E})$ defined by $\mathcal{V} = \mathrm{me}(c)$ and $\mathcal{E} = \{\{\mathcal{P}_1, \mathcal{P}_2\} \in \mathcal{V} \times \mathcal{V} \mid \mathcal{P}_1 \neq \mathcal{P}_2, \bigcup \mathrm{S}^\star(\mathcal{P}_1) \cap \bigcup \mathrm{S}^\star(\mathcal{P}_2) \neq \emptyset\}$ is acyclic (i.e., multiple inheritance does not cyclically connect partition clusters).

W4 For every class $c \in \mathcal{C}$: if the graph $G_{\mathbf{S}}(c)$ has an edge, i.e., distinct clusters $\mathcal{P}_1, \mathcal{P}_2 \in \mathrm{me}(c)$ have a common subclass, then every path from a subclass of c to the top element of $(\mathcal{C}, \Rightarrow)$ goes through c (that is, multiple inheritance can be *locally isolated* in the graph $(\mathcal{C}, \Rightarrow)$).

Informally, these conditions restrict multiple inheritance in a way which ensures that a model for the schema S can be built bottom up from models of subschemas. Well-structuredness does not appear to be very restrictive in practice. For instance, let us reconsider the Plant Example.

Example 5.7 (Plant Example: well-structured POB-schema) The POB-schema S given in Example 5.4 is well-structured:

- Every class is a subclass of plants.
- The classes annuals_herbs and perennials_flowers are t-disjoint.
- There are no cyclically connected partition clusters.
- The multiple inheritance at the classes annuals_herbs and perennials_flowers is locally isolated under the class plants.

As far as well-structured POB-schemas are concerned, we have the nice result that pseudo-consistency is a necessary and sufficient condition for consistency. However, the proof of this result is highly nontrivial (see [11]).

Theorem 5.3 *Every pseudo-consistent and well-structured POB-schema* $\mathbf{S}$ *is consistent.*

It is easily seen that any $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ without multiple inheritance, i.e., $|\{d \in \mathcal{C} \mid c \Rightarrow d\}| \leq 1$ for each class $c \in \mathcal{C}$, satisfies W2-W4. We obtain the following corollary to Theorem 5.3.

Corollary 5.4 *Every POB-schema with top element and without multiple inheritance is consistent.*

It now remains to show that the pseudo-consistency and the well-structuredness of a POB-schema can be decided efficiently. We first concentrate on the pseudo-consistency. Algorithm 5.2 uses Algorithm 5.1 to provide a procedure to check pseudo-consistency.

Theorem 5.5 *Deciding whether a given POB-schema* $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ *is pseudo-consistent is can be done using Algorithm 5.2 in time* $O(n(e+n))$*, where* $n = |\mathcal{C}|$ *and* e *is the number of directed edges in* $(\mathcal{C}, \Rightarrow)$.

Proof. Algorithm 5.2 decides the pseudo-consistency of $\mathbf{S}$ in time $O(ne)$. It uses Algorithm 5.1, which computes the reachability relation of the graph $(\mathcal{C}, \Rightarrow)$ in time $O(n(e+n))$.

We first show that Algorithm 5.1 runs in time $O(n(e+n))$: The initialization steps 1–3, 4–7, and 8 run in time $O(n^2)$, $O(ne)$, and $O(n)$, respectively. Next, it is easy to see that the **for**-loop in 15–17 is performed as many times as there are edges in $(\mathcal{C}, \Rightarrow)$, and each execution takes $O(n)$ time. Thus, the whole **while**-loop in 9–19 runs in time $O(ne)$.

Hence, also Algorithm 5.2 runs in time $O(n(e+n))$: The steps 1–2 run in time $O(n(e+n))$. The **for**-loop in 4–5 runs in linear time in the input size of me (i.e, in e). Thus, the whole **for**-loop in 3–5 runs in time $O(ne)$. $\Box$

We next focus on deciding well-structuredness via Algorithm 5.3.

Theorem 5.6 *The problem of deciding whether a pseudo-consistent POB-schema* $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ *is well-structured can be solved using Algorithm 5.3 in time* $O(n^2 e)$*, where* $n = |\mathcal{C}|$ *and* e *is the number of directed edges in* $(\mathcal{C}, \Rightarrow)$.

Proof. Algorithm 5.3 decides the well-structuredness of $\mathbf{S}$. The steps 1–3 check whether $\mathbf{S}$ satisfies W1. In 4–16, it is then checked whether $\mathbf{S}$ satisfies W2. Moreover, the union of all undirected graphs $G_{\mathbf{S}}(c)$ with $c \in \mathcal{C}$ and the set of all classes with multiple inheritance at subclasses are computed. Step 17 checks that all the graphs $G_{\mathbf{S}}(c)$ with $c \in \mathcal{C}$ are acyclic (W2 ensures that the $G_{\mathbf{S}}(c)$ have disjoint edge sets). In 18–22, it is finally checked whether $\mathbf{S}$ satisfies W4.

We now show that Algorithm 5.3 runs in time $O(n^2 e)$. It is easy to see that the steps 1–2, 3, and 4 run in time $O(ne)$, $O(n)$, and $O(n(e+n)) = O(ne)$, respectively (note that W1 ensures $e \geq n-1$). Step 10 is done one time for each edge in $(\mathcal{C}, \Rightarrow)$ and each class in a set of classes limited by $\mathcal{C}$. The set $\mathcal{D}$ there can be computed in time $O(n)$. The tests in Step 11 and 12 can be done, using a simple algorithm, in time $O(n)$. Hence, the steps 5–16 run in time $O(n^2 e)$. In step 17, the number of clusters in $\bigcup \mathrm{me}(\mathcal{C})$ is in the worst case equal to e. Thus, step 17 can be performed in time $O(e)$

Algorithm 5.1: reachability(S)

Input: POB-schema $\mathrm{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$.
Output: If S does not satisfy P1, then *nil* is returned. Otherwise, a mapping $w\colon \mathcal{C} \times \mathcal{C} \to [0,1]$ is returned such that $w(c,d)$ is the product of the edge probabilities on all paths from c up to d if such a path exists and $w(c,d)$ is 0 otherwise.

for each $c, d \in \mathcal{C}$ **do**
  if $c = d$ **then** $w(c,d) := 1$
    else $w(c,d) := 0$;
for each $c \in \mathcal{C}$ **do begin**
  $\mathrm{S}(c) := \{d \in \mathcal{C} \mid d \Rightarrow c\}$;
  $\delta(c) := |\{d \in \mathcal{C} \mid c \Rightarrow d\}|$
end;
$N := \{c \in \mathcal{C} \mid \delta(c) = 0\}$;
while $N \neq \emptyset$ **do begin**
  take any $c \in N$;
  $N := N - \{c\}$;
  for each $d \in \mathrm{S}(c)$ **do begin**
    $\delta(d) := \delta(d) - 1$;
    if $\delta(d) = 0$ **then** $N := N \cup \{d\}$;
    for each $e \in \mathcal{C}$ with $w(c,e) > 0$ **do**
      if $w(d,e) > 0$ **and** $w(d,e) \neq w(d,c) \cdot w(c,e)$ **then return** *nil*
        else $w(d,e) := w(d,c) \cdot w(c,e)$
  end;
end;
return w.

Algorithm 5.2: pseudo-consistent(S)

Input: POB-schema $\mathrm{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$.
Output: *true* if S is pseudo-consistent and *false* otherwise.

$w := \mathit{reachability}(\mathbf{S})$;
if $w = \mathit{nil}$ **then return** *false*; (S does not satisfy P1)
for each $c \in \mathcal{C}$ **do**
  for each $\mathcal{P} \in \bigcup \mathrm{me}(\mathcal{C})$ **do**
    if $|\{e \in \mathcal{P} \mid w(c,e) > 0\}| > 1$ **then return** *false*; (S does not satisfy P2)
return *true*. (S is pseudo-consistent)

Algorithm 5.3: well-structured(S)

Input: Pseudo-consistent POB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$.
Output: *true* if S is well-structured and *false* otherwise.
Notation: We use $top(\mathbf{S})$ to denote the top element of $(\mathcal{C}, \Rightarrow)$. For classes $c \in \mathcal{C}$, the expression $\mathbf{S} - c$ denotes the POB-schema that is obtained from S by removing c. $\max(\mathcal{D})$, where $\mathcal{D} \subseteq \mathcal{C}$, is the set of all maximal members in $\mathcal{D}$ w.r.t. $\Rightarrow^\star$

1. **for each** $c \in \mathcal{C}$ **do**
2. $\delta(c) := |\{d \in \mathcal{C} \mid c \Rightarrow d\}|$;
3. **if** $|\{c \in \mathcal{C} \mid \delta(c) = 0\}| > 1$ **then return** *false*; (S does not satisfy W1)
4. $w := reachability(\mathbf{S})$;
5. $\mathcal{E} := \emptyset$;
6. $M := \emptyset$;
7. **for each** $c \in \mathcal{C}$ **do**
8.   **for each** distinct $\mathcal{P}_1, \mathcal{P}_2 \in \mathrm{me}(c)$ **do**
9.   **for each** $(c_1, c_2) \in \mathcal{P}_1 \times \mathcal{P}_2$ **do begin**
10. $\mathcal{D} := \{d \in \mathcal{C} \mid w(d, c_1) > 0,\ w(d, c_2) > 0\}$;
11.   **if** $|\max(\mathcal{D})| > 1$ or $\mathcal{D} \cap \{c_1, c_2\} \neq \emptyset$ **then return** *false* (S does not satisfy W2)
12.   **else if** $|\max(\mathcal{D})| = 1$ **then begin**
13. $\mathcal{E} := \mathcal{E} \cup \{\{\mathcal{P}_1, \mathcal{P}_2\}\}$;
14. $M := M \cup \{c\}$
15.   **end**
16.   **end**;
17. **if** $(\bigcup \mathrm{me}(\mathcal{C}), \mathcal{E})$ contains a cycle **then return** *false*; (S does not satisfy W3)
18. **for each** $c \in M$ **do begin**
19. $v := reachability(\mathbf{S} - c)$;
20.   **for each** $d \in \mathcal{C}$ with $w(d, c) > 0$ **do**
21.   **if** $v(d, top(\mathbf{S})) > 0$ **then return** *false*; (S does not satisfy W4)
22. **end**;
23. **return** *true*. (S is well-structured)

using standard algorithms for checking acyclicity. Finally, it is easy to see that the steps 18–22 run in time $O(n^2 e)$. $\Box$

6 Inheritance and Probabilistic Object Base Instances

Thus far, we have not addressed inheritance of attributes which may arise through subclass relationships in a POB-schema S. For example, if c is a subclass of d, and d's type has an attribute A, then the class c should inherit this attribute, unless c has already such an attribute. The issue of inheritance, and in particular of multiple inheritance, has been extensively discussed in the literature, e.g. [4]. We next incorporate inheritance in our framework, and finally define instances of a POB-schema.

6.1 Inheritance completion and fully inherited schemas

We assume that any schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ has an associated inheritance strategy, $\mathsf{inh}_{\mathbf{S}}$, that determines from which superclass d (if any) a class c inherits a top-level attribute A. More formally, $\mathsf{inh}_{\mathbf{S}} : \mathcal{C} \times \mathbf{A} \to \mathcal{C}$, where $\mathbf{A}$ is the set of all top-level attributes of $\mathbf{A}$, is a partial mapping that assigns each pair of a class $c \in \mathcal{C}$ and a top-level attribute $A \in \mathbf{A}$, a closest class d such that (i) $c \Rightarrow^\star d \Rightarrow^\star d'$ for some class d' and (ii) $A \in \mathbf{B}$ where $\sigma(d')$ is a tuple type over a set of attributes $\mathbf{B}$; here, "closest" means that no proper subclass $d'' \in S^\star(d) - \{d\}$ with properties (i) and (ii) exists. (In particular, $\mathsf{inh}_{\mathbf{S}}(c, A) = c$ if $\sigma(c)$ possesses A.) The value of $\mathsf{inh}_{\mathbf{S}}(c, A)$ is undefined, if no such d exists.

This notion of an inheritance strategy covers strategies (such as an ordering on classes) that are commonly used to resolve multiple inheritance in practice. Similarly, if we wish to use the strategy of the O2 system [2] where renamed inheritance of the same attribute with distinct origins is desired, we could generalize $\mathsf{inh}_{\mathbf{S}}(c, A)$ to return all pairs d, A' of classes d from which attribute A, renamed to A', is inherited.

Applying $\mathsf{inh}_{\mathbf{S}}$ on a POB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ induces another POB-schema $\mathbf{S}' = (\mathcal{C}', \sigma', \Rightarrow', \mathrm{me}', \wp')$, which only differs from $\mathbf{S}$ in its type assignment σ': $\sigma'(c)$ is the tuple type $\tau = [A_1 \colon \tau_1, \ldots, A_k \colon \tau_k]$ where $A_1, \ldots, A_k$ are all attributes that are inherited from c via $\mathsf{inh}_{\mathbf{S}}$ from classes $d_1, \ldots, d_k$, respectively, and $\tau_1, \ldots, \tau_k$ are the types of the attributes in $\sigma(\tau_1), \ldots, \sigma(\tau_k)$, respectively. We call this schema $\mathbf{S}'$ the *inheritance completion* of $\mathbf{S}$, and a scheme $\mathbf{S}$ which equals $\mathbf{S}'$ *fully inherited*.

Example 6.1 (Plant Example: probabilistic object base schema) Let us review the schema $\mathbf{S}$ for the Plant Example as defined in Example 5.4. It is easily checked that for every subclass c of any class d, $\sigma(d)$ is a type over a subset of the attributes of $\sigma(c)$, i.e., all attributes in d are already present in c. Thus, no attributes are inherited from proper superclasses, which means that $\mathbf{S}$ is fully inherited. The type assignment σ in $\mathbf{S}$ may be considered ill-designed, however, since natural inheritance relationships are not reflected in it.

Consider the redesigned type assignment σ' in Table 3, and adopt an inheritance strategy $\mathsf{inh}_{\mathbf{S}}$ which resolves multiple inheritance by ordering "left-to-right" in Figure 1, i.e., orders annuals before herbs and perennials before flowers.[2] Then, the inheritance completion of the redesigned schema $\mathbf{S}' = (\mathcal{C}, \sigma', \Rightarrow, \mathrm{me}, \wp)$ is the original schema $\mathbf{S}$.

In the rest of this paper, we implicitly assume that schemas $\mathbf{S}$ are consistent and fully inherited. This applies in particular to the definition of POB-instance in the next subsection, and the definitions of the operations in the POB-Algebra in Section 7. Extending the definitions to schemas $\mathbf{S}$ which are not full inherited–by replacing $\mathbf{S}$ with its inheritance completion $\mathbf{S}'$ is straightforward.

[2] No renaming is assumed here for the same attribute with distinct origins.

c	$\sigma(c)$
plants	[pname : string, soil : soiltype, rain : integer]
annuals	[sun : suntype]
perennials	[sun : suntype, expyears : integer]
vegetables	[sun : suntype, expyears : integer]
herbs	[sun : suntype, expyears : integer, classification : string]
flowers	[sun : suntype, expyears : integer, classification : string]
annuals_herbs	[ ]
perennials_flowers	[ ]

Table 3: Redesigned type assignment σ'

6.2 Probabilistic object base instance

We are now ready to define a probabilistic object base instance (POB-instance). The following assumption is common in the context of object-oriented databases[22].

Assumption. In the rest of this paper, we assume that there is a (countably) infinite set $\mathcal{O}$ of *object identifiers* (*oids*).

Each object, represented by an oid, is associated with a value. The objects populate a POB-instance as follows.

Definition 6.1 (probabilistic object base instance) Let $\mathrm{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ be a consistent POB-schema. A *probabilistic object base instance (POB-instance)* over S is a pair (π, ν), where:

- $\pi : \mathcal{C} \rightarrow 2^{\mathcal{O}}$ maps each class c to a finite subset of $\mathcal{O}$, such that $\pi(c_1) \cap \pi(c_2) = \emptyset$ for different $c_1, c_2 \in \mathcal{C}$. That is, the classes in $\mathcal{C}$ are mapped to pairwise disjoint sets of oids. We use $\pi(\mathcal{C})$ to abbreviate $\bigcup\{\pi(c) \mid c \in \mathcal{C}\}$. We define the mapping $\pi^\star : \mathcal{C} \rightarrow 2^{\mathcal{O}}$ by $\pi^\star(c) = \bigcup\{\pi(c') \mid c' \in \mathcal{C},\, c' \Rightarrow^\star c\}$.

  Intuitively, $\pi(c)$ denotes the ids of all objects that are *defined in* the class c, while $\pi^\star(c)$ denotes the ids of all objects that *belong to* the class c.

- ν maps each oid $o \in \pi(\mathcal{C})$ to a probabilistic value of the appropriate type, i.e., type $\sigma(c)$ for the class c such that $o \in \pi(c)$.

Let us provide a POB-instance for the POB-schema of Example 5.4.

Example 6.2 (Plant Example: probabilistic object base instance) A POB-instance over the POB-schema shown in Example 5.4 is given as follows:

- π and $\pi^\star$ are the following mappings:

c	$\pi(c)$	$\pi^\star(c)$
plants	$\{o_1\}$	$\{o_1, o_2, o_3, o_4, o_5, o_6, o_7\}$
annuals	$\{\}$	$\{o_2, o_3, o_5, o_6, o_7\}$
perennials	$\{\}$	$\{o_4\}$
vegetables	$\{\}$	$\{\}$
herbs	$\{\}$	$\{o_2, o_3, o_5, o_6, o_7\}$
flowers	$\{\}$	$\{o_4\}$
annuals_herbs	$\{o_2, o_3, o_5, o_6, o_7\}$	$\{o_2, o_3, o_5, o_6, o_7\}$
perennials_flowers	$\{o_4\}$	$\{o_4\}$

Clearly, this is a very simple probabilistic object base (it contains only seven distinct objects).

- ν is the mapping shown in Table 4.

oid	$\nu(oid)$
o_1	$[\mathsf{pname}: \langle\{\mathsf{Lady\text{-}Fern}, \mathsf{Ostrich\text{-}Fern}\}, \mathrm{u}, \mathrm{u}\rangle$, $\mathsf{soil}: \langle\{\mathsf{loamy}\}, \mathrm{u}, \mathrm{u}\rangle$, $\mathsf{rain}: \langle\{25, \ldots, 30\}, \mathrm{u}, \mathrm{u}\rangle]$
o_2	$[\mathsf{pname}: \langle\{\mathsf{Cuban\text{-}Basil}, \mathsf{Lemon\text{-}Basil}\}, \mathrm{u}, \mathrm{u}\rangle$, $\mathsf{soil}: \langle\{\mathsf{loamy}, \mathsf{sandy}\}, 0.7\,\mathrm{u}, 1.3\,\mathrm{u}\rangle$, $\mathsf{rain}: \langle\{20, \ldots, 30\}, \mathrm{u}, \mathrm{u}\rangle$, $\mathsf{sun}: \langle\{\mathsf{mild}, \mathsf{medium}\}, 0.8\,\mathrm{u}, 1.2\,\mathrm{u}\rangle$, $\mathsf{expyears}: \langle\{2, 3, 4\}, 0.6\,\mathrm{u}, 1.8\,\mathrm{u}\rangle$, $\mathsf{classification}: \langle\{\mathsf{french}, \mathsf{silver}, \mathsf{wooly}\}, 0.6\,\mathrm{u}, 1.8\,\mathrm{u}\rangle]$
o_3	$[\mathsf{pname}: \langle\{\mathsf{Mint}\}, \mathrm{u}, \mathrm{u}\rangle$, $\mathsf{soil}: \langle\{\mathsf{loamy}\}, \mathrm{u}, \mathrm{u}\rangle$, $\mathsf{rain}: \langle\{20\}, \mathrm{u}, \mathrm{u}\rangle$, $\mathsf{sun}: \langle\{\mathsf{mild}\}, \mathrm{u}, \mathrm{u}\rangle$, $\mathsf{expyears}: \langle\{2, 3, 4\}, 0.6\,\mathrm{u}, 1.8\,\mathrm{u}\rangle$, $\mathsf{classification}: \langle\{\mathsf{french}, \mathsf{silver}, \mathsf{wooly}\}, 0.6\,\mathrm{u}, 1.8\,\mathrm{u}\rangle]$
o_4	$[\mathsf{pname}: \langle\{\mathsf{Aster}, \mathsf{Salvia}\}, \mathrm{u}, \mathrm{u}\rangle$, $\mathsf{soil}: \langle\{\mathsf{loamy}, \mathsf{sandy}\}, 0.6\,\mathrm{u}, 1.4\,\mathrm{u}\rangle$, $\mathsf{rain}: \langle\{20, \ldots, 25\}, \mathrm{u}, \mathrm{u}\rangle$, $\mathsf{sun}: \langle\{\mathsf{mild}\}, \mathrm{u}, \mathrm{u}\rangle$, $\mathsf{expyears}: \langle\{2, 3, 4\}, 0.6\,\mathrm{u}, 1.8\,\mathrm{u}\rangle$, $\mathsf{classification}: \langle\{\mathsf{french}, \mathsf{silver}, \mathsf{wooly}\}, 0.6\,\mathrm{u}, 1.8\,\mathrm{u}\rangle]$
o_5	$[\mathsf{pname}: \langle\{\mathsf{Thyme}\}, \mathrm{u}, \mathrm{u}\rangle$, $\mathsf{soil}: \langle\{\mathsf{loamy}\}, \mathrm{u}, \mathrm{u}\rangle$, $\mathsf{rain}: \langle\{20, \ldots, 25\}, \mathrm{u}, \mathrm{u}\rangle$, $\mathsf{sun}: \langle\{\mathsf{mild}, \mathsf{medium}\}, 0.8\,\mathrm{u}, 1.2\,\mathrm{u}\rangle$, $\mathsf{expyears}: \langle\{2, 3\}, 0.8\,\mathrm{u}, 1.2\,\mathrm{u}\rangle$, $\mathsf{classification}: \langle\{\mathsf{french}, \mathsf{silver}, \mathsf{wooly}\}, 0.6\,\mathrm{u}, 1.8\,\mathrm{u}\rangle]$
o_6	$[\mathsf{pname}: \langle\{\mathsf{Mint}\}, \mathrm{u}, \mathrm{u}\rangle$, $\mathsf{soil}: \langle\{\mathsf{loamy}\}, \mathrm{u}, \mathrm{u}\rangle$, $\mathsf{rain}: \langle\{20\}, \mathrm{u}, \mathrm{u}\rangle$, $\mathsf{sun}: \langle\{\mathsf{mild}\}, \mathrm{u}, \mathrm{u}\rangle$, $\mathsf{expyears}: \langle\{2, 3, 4\}, 0.6\,\mathrm{u}, 1.8\,\mathrm{u}\rangle$, $\mathsf{classification}: \langle\{\mathsf{apple}, \mathsf{curly}\}, 0.6\,\mathrm{u}, 1.4\,\mathrm{u}\rangle]$
o_7	$[\mathsf{pname}: \langle\{\mathsf{Sage}\}, \mathrm{u}, \mathrm{u}\rangle$, $\mathsf{soil}: \langle\{\mathsf{sandy}\}, \mathrm{u}, \mathrm{u}\rangle$, $\mathsf{rain}: \langle\{20, 21\}, \mathrm{u}, \mathrm{u}\rangle$, $\mathsf{sun}: \langle\{\mathsf{mild}\}, \mathrm{u}, \mathrm{u}\rangle$, $\mathsf{expyears}: \langle\{2, 3, 4\}, 0.6\,\mathrm{u}, 1.8\,\mathrm{u}\rangle$, $\mathsf{classification}: \langle\{\mathsf{red}, \mathsf{tricolor}\}, 0.6\,\mathrm{u}, 1.4\,\mathrm{u}\rangle]$

Table 4: Value assignment ν

In classical object bases, the extent of a class c consists of all oids belonging to c. In probabilistic object bases, the probabilistic extent of c specifies the probability that an oid belongs to c.

Definition 6.2 (probabilistic extent) Let $\mathbf{I} = (\pi, \nu)$ be a POB-instance over the consistent POB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$. For all classes $c \in \mathcal{C}$, the *probabilistic extent* of c, denoted $\mathrm{ext}(c)$, maps each oid $o \in \pi(\mathcal{C})$ to a *set* of rational numbers in $[0, 1]$ as follows:

1. If $o \in \pi^\star(c)$, then $\mathrm{ext(c)}(o) = \{1\}$.
2. If $o \in \pi^\star(c')$ with a class $c' \in \mathcal{C}$ that is t-disjoint from c (that is, for all models ε of $\mathbf{S}$, the sets $\varepsilon(c')$ and $\varepsilon(c)$ are disjoint), then $\mathrm{ext(c)}(o) = \{0\}$.
3. Otherwise, $\mathrm{ext(c)}(o) = \{p \mid p$ is the product of the edge probabilities on a path from c up to a class $c' \in \mathcal{C}$, where c' is minimal with $o \in \pi^\star(c')$ and $c \Rightarrow^\star c'$ $\}$.

Let us return to the Plant Example to see what the extents of the various classes are.

Example 6.3 (Plant Example: probabilistic extent) Let us consider the probabilistic extents of the classes $\mathsf{annuals\_herbs}$ and $\mathsf{perennials\_flowers}$ in the Plant Example:

$$
\begin{array}{ll}
\mathrm{ext}(\mathsf{annuals\_herbs})(o_1) = \{0.24\} & \mathrm{ext}(\mathsf{perennials\_flowers})(o_1) = \{0.12\} \\
\mathrm{ext}(\mathsf{annuals\_herbs})(o_2) = \{1\} & \mathrm{ext}(\mathsf{perennials\_flowers})(o_2) = \{0\} \\
\mathrm{ext}(\mathsf{annuals\_herbs})(o_3) = \{1\} & \mathrm{ext}(\mathsf{perennials\_flowers})(o_3) = \{0\} \\
\mathrm{ext}(\mathsf{annuals\_herbs})(o_4) = \{0\} & \mathrm{ext}(\mathsf{perennials\_flowers})(o_4) = \{1\} \\
\mathrm{ext}(\mathsf{annuals\_herbs})(o_5) = \{1\} & \mathrm{ext}(\mathsf{perennials\_flowers})(o_5) = \{0\} \\
\mathrm{ext}(\mathsf{annuals\_herbs})(o_6) = \{1\} & \mathrm{ext}(\mathsf{perennials\_flowers})(o_6) = \{0\} \\
\mathrm{ext}(\mathsf{annuals\_herbs})(o_7) = \{1\} & \mathrm{ext}(\mathsf{perennials\_flowers})(o_7) = \{0\}
\end{array}
$$

Definition 6.3 (coherent POB-instance) Let $\mathbf{I} = (\pi, \nu)$ be a POB-instance over the consistent POB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$. The POB-instance $\mathbf{I}$ is *coherent* iff for all classes $c \in \mathcal{C}$ and all objects $o \in \pi(\mathcal{C})$, the probabilistic extent $\mathrm{ext(c)}(o)$ contains *at most* one element.

It is easy to see that the Plant Example described thus far is coherent. Note that testing whether a given POB-instance $\mathbf{I}$ of a consistent schema $\mathbf{S}$ is coherent is feasible in polynomial time.

7 Probabilistic Object Bases: Algebraic Operations

In this section, we formally define the analogs of the classical relational operations on probabilistic object bases. In the relational model, all standard relational operations take relations as input (perhaps with other inputs as well) and produce relations as output. In the same vein, all standard operations on POBs take POB-instances as input, and produce POB-instances as output. Recall that all POB-schemas of input POB-instances are implicitly assumed to be consistent and fully inherited.

7.1 Selection

The first important operation to be defined is *selection*. Intuitively, given a POB-instance $\mathbf{I}$ over the POB-schema $\mathbf{S}$, the result of a selection operation is another POB-instance $\mathbf{I}'$ over $\mathbf{S}$ such that the objects in the extents of the classes in $\mathbf{I}'$ all satisfy the selection condition of the query.

Before describing the selection operation, we must formally define the syntax and the semantics of probabilistic selection conditions. We start by defining path expressions.

Definition 7.1 (path expression) Let $\tau = [A_1 : \tau_1, \ldots, A_k : \tau_k]$ be any type. Then, (i) every A_i is a path expression for τ; (ii) if τ_i is a tuple type and P_i is a path expression for τ_i, then $A_i.P_i$ is a path expression for τ, for every $i = 1, \ldots, k$.

We now define the syntax of atomic selection conditions.

Definition 7.2 (atomic selection condition) Let S $= (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ be a POB-schema and let $\mathcal{X}$ be a set of *object variables*. An *atomic selection condition* has one of the following forms:

- $x \in c$, where x is an object variable from $\mathcal{X}$ and c is a class from $\mathcal{C}$.
- $x.P\ \theta\ v$, where x is an object variable from $\mathcal{X}$, P is a path expression over attributes names from $\mathcal{A}$, θ is a binary predicate from $\{=, \neq, \leq, \geq, <, >, \subseteq, \supseteq, \in, \ni\}$, and v is a value.
- $x.P_1 =_{\otimes} x.P_2$, where x is an object variable from $\mathcal{X}$, P_1 and P_2 are two different path expressions over attributes from $\mathcal{A}$, and $\otimes$ is a probabilistic conjunction strategy.

Let us consider some examples of atomic selection conditions.

Example 7.1 (Plant Example: atomic selection condition) In the Plant Example, some atomic selection conditions are given as follows (x is an object variable):

- Find all objects *that are annuals and herbs.*
  This selection can be expressed by the atomic selection condition $x \in \mathsf{annuals\_herbs}$.
- Find all objects *that require a mild sun.*
  This selection can be expressed by the atomic selection condition $x.\mathsf{sun} = \mathsf{mild}$.
- Find all objects *that require over 21 units of rain.*
  This selection can be expressed by the atomic selection condition $x.\mathsf{rain} > 21$.

We now define the syntax of selection conditions.

Definition 7.3 (selection condition) Let S $= (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ be a POB-schema. We define *conjunctive* and *disjunctive selection conditions* by induction as follows.

If ϕ is an atomic selection condition and $\otimes$ is a probabilistic conjunction strategy, then ϕ is a conjunctive selection condition over $\otimes$. If ϕ and ψ are conjunctive selection conditions over the same object variable and the same probabilistic conjunction strategy $\otimes$, then $\phi \otimes \psi$ is a conjunctive selection condition over $\otimes$.

If ϕ is an atomic selection condition and $\oplus$ is a probabilistic disjunction strategy, then ϕ is a disjunctive selection condition over $\oplus$. If ϕ and ψ are disjunctive selection conditions over the same object variable and the same probabilistic disjunction strategy $\oplus$, then $\phi \oplus \psi$ is a disjunctive selection condition over $\oplus$. A *selection condition* is a conjunctive or disjunctive selection condition.

Let us illustrate this definition via the Plant Example.

Example 7.2 (Plant Example: selection condition) In the Plant Example, some selection conditions are given as follows (x is an object variable):

- The atomic selection conditions $x \in \mathsf{annuals\_herbs}$, $x.\mathsf{sun} = \mathsf{mild}$, and $x.\mathsf{rain} > 21$ given in Example 7.1 are selection conditions.
- Find all objects *that are annuals and herbs and that require a mild sun.* This selection can be expressed by the selection condition $x \in \mathsf{annuals\_herbs} \otimes x.\mathsf{sun} = \mathsf{mild}$, where $\otimes$ is a probabilistic conjunction strategy.
- Find all objects *that require a mild sun or over 21 units of rain.* This selection can be expressed by the selection condition $x.\mathsf{sun} = \mathsf{mild} \oplus x.\mathsf{rain} > 21$, where $\oplus$ is a probabilistic disjunction strategy.

We are now ready to define the syntax of probabilistic selection conditions.

Definition 7.4 (probabilistic selection condition) Let $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ be a POB-schema. (i) If ϕ is a selection condition and L and U are real numbers from $[0, 1]$ with $L \leq U$, then $(\phi)[L, U]$ is a probabilistic selection condition. (ii) If ϕ and ψ are probabilistic selection conditions *over the same object variable*, then $\neg\phi$, $(\phi \wedge \psi)$, and $(\phi \vee \psi)$ are probabilistic selection conditions.

Let us consider some examples of probabilistic selection conditions.

Example 7.3 (Plant Example: probabilistic selection condition) In the Plant Example, some probabilistic selection conditions are given as follows (x is an object variable):

- The selection of all objects *that require both a mild sun and over 21 units of rain with a probability of 30–50%*, can be done by using the probabilistic selection condition $(x.\mathsf{sun} = \mathsf{mild} \otimes x.\mathsf{rain} > 21)[0.3, 0.5]$, where $\otimes$ is a probabilistic conjunction strategy.
- The selection of all objects *that require a mild sun with a probability of at least 40%,* and *over 21 units of rain with a probability of at least 80%,* can be done by using the probabilistic selection condition $(x.\mathsf{sun} = \mathsf{mild})[0.4, 1] \wedge (x.\mathsf{rain} > 21)[0.8, 1]$.

It is important to note that each selection condition and each probabilistic selection condition contains exactly one object variable.

It now remains to define the semantics of selection and probabilistic selection conditions. For this purpose, each pair $(\mathbf{S}, o)$ consisting of a POB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ and an oid $o \in \pi(\mathcal{C})$ in a POB-instance $\mathbf{I} = (\pi, \nu)$ over $\mathbf{S}$ is associated with a probabilistic interpretation $\mathrm{prob}_{\mathbf{I},o}$, which assigns a probabilistic interval to selection conditions, and a truth value to probabilistic selection conditions. We start by assigning probabilistic intervals to atomic selection conditions:

Definition 7.5 (satisfaction of atomic selection conditions) Let $\mathbf{I} = (\pi, \nu)$ be a POB-instance over the POB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ and let $o \in \pi(\mathcal{C})$. The *probabilistic interpretation* with respect to $\mathbf{I}$ and o, denoted $\mathrm{prob}_{\mathbf{I},o}$, is the partial mapping from all atomic selection conditions to the set of all subintervals of $[0,1]$ that is defined by:

- $\mathrm{prob}_{\mathbf{I},o}(x \in c) = [\min(\mathrm{ext(c)}(o)), \max(\mathrm{ext(c)}(o))]$.
- If $\nu(o).A = \langle V, \alpha, \beta\rangle$ and AP is a path expression for the type of o, then

$$\mathrm{prob}_{\mathbf{I},o}(x.AP\,\theta\,v) = [L, \min(1,U)] \;\text{ such that }\; [L,U] = \textstyle\sum_{u\in W}[\alpha(u), \beta(u))]\,,$$

  where $W = \{u \in V \mid val(P,u)\,\theta\,v\}$ and $val(P,u)$ is defined as follows:[3] $val(P,u) = u$, if P is empty, and $val(P,u) = val(P',u')$ if $P = .B\,P'$ and $u.B = u'$ otherwise (i.e., $val(P,u)$ returns the value of the component of $\nu(o)$ described by the path expression $A\,P$). Note that we canonically define $\mathrm{prob}_{\mathbf{I},o}(x.A\,\theta\,v) = [0,0]$ for $\{u \in V \mid u\,\theta\,v\} = \emptyset$. We further assume that $\mathrm{prob}_{\mathbf{I},o}(x.AP\,\theta\,v)$ is undefined if the path expression AP is undefined for $\nu(o)$, or if $val(P,u)\,\theta\,v$ is undefined for some $u \in V$.

- If $\nu(o).A_i = \langle V_i, \alpha_i, \beta_i\rangle$ and A_iP_i is a path expression for the type of o, for $i \in \{1,2\}$, then:

$$\mathrm{prob}_{\mathbf{I},o}(x.A_1\,P_1 =_\otimes x.A_2\,P_2) = [L, \min(1,U)] \;\text{ such that}$$
$$[L,U] = \textstyle\sum_{(u_1,u_2)\in W}[\alpha_1(u_1), \beta_1(u_1)] \otimes [\alpha_2(u_2), \beta_2(u_2)]\,,$$

  where $W = \{(u_1,u_2) \in V_1 \times V_2 \mid val(P_1,u_1) = val(P_2,u_2)\}$, and $val(\cdot,\cdot)$ is defined as above. We canonically define $\mathrm{prob}_{\mathbf{I},o}(x.A_1 =_\otimes x.A_2) = [0,0]$ for an empty sum. We assume that $\mathrm{prob}_{\mathbf{I},o}(x.A_1 =_\otimes x.A_2)$ is undefined if $A_1\,P_1$ or $A_2\,P_2$ is undefined for $\nu(o)$.

Let us give an example to illustrate this definition.

Example 7.4 (Plant Example: satisfaction of atomic selection conditions) In the Plant Example, the probabilistic interpretations $\mathrm{prob}_{\mathbf{I},o}$ with $o \in \{o_1, o_2, \ldots, o_7\}$ map the atomic selection conditions $x \in \mathsf{annuals\_herbs}$, $x.\mathsf{sun} = \mathsf{mild}$, and $x.\mathsf{rain} > 21$ to the following subintervals of $[0,1]$:

[3]As usual, the sum $\sum_{x\in X}[\alpha(x), \epsilon(x)]$ denotes $[\sum_{x\in X}\alpha(x), \sum_{x\in X}\epsilon(x)]$.

o	$\mathrm{prob}_{\mathbf{I},o}(x \in \mathsf{annuals\_herbs})$	$\mathrm{prob}_{\mathbf{I},o}(x.\mathsf{sun} = \mathsf{mild})$	$\mathrm{prob}_{\mathbf{I},o}(x.\mathsf{rain} > 21)$
o_1	$[0.24, 0.24]$	undefined	$[1.00, 1.00]$
o_2	$[1.00, 1.00]$	$[0.40, 0.60]$	$[0.82, 0.82]$
o_3	$[1.00, 1.00]$	$[1.00, 1.00]$	$[0.00, 0.00]$
o_4	$[0.00, 0.00]$	$[1.00, 1.00]$	$[0.67, 0.67]$
o_5	$[1.00, 1.00]$	$[0.40, 0.60]$	$[0.67, 0.67]$
o_6	$[1.00, 1.00]$	$[1.00, 1.00]$	$[0.00, 0.00]$
o_7	$[1.00, 1.00]$	$[1.00, 1.00]$	$[0.00, 0.00]$

We next assign probabilistic intervals to selection conditions:

Definition 7.6 (satisfaction of selection conditions) Let $\mathbf{I} = (\pi, \nu)$ be a POB-instance over the POB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ and let $o \in \pi(\mathcal{C})$. We extend $\mathrm{prob}_{\mathbf{I},o}$ to a partial mapping from the set of all selection conditions to the set of all closed subintervals of [0,1] as follows:

$$\begin{aligned}\mathrm{prob}_{\mathbf{I},o}(\phi \otimes \psi) &= \mathrm{prob}_{\mathbf{I},o}(\phi) \otimes \mathrm{prob}_{\mathbf{I},o}(\psi).\\ \mathrm{prob}_{\mathbf{I},o}(\phi \oplus \psi) &= \mathrm{prob}_{\mathbf{I},o}(\phi) \oplus \mathrm{prob}_{\mathbf{I},o}(\psi).\end{aligned}$$

Let us illustrate this definition via the Plant Example.

Example 7.5 (Plant Example: satisfaction of selection conditions) In the Plant Example, the two selection conditions ϕ_{st} = "$x \in \mathsf{annuals\_herbs} \otimes_{st} x.\mathsf{sun} = \mathsf{mild}$" and ψ_{st} = "$x.\mathsf{sun} = \mathsf{mild} \otimes_{st} x.\mathsf{rain} > 21$" are assigned the following subintervals of $[0, 1]$:

o	$\mathrm{prob}_{\mathbf{I},o}(\phi_{in})$	$\mathrm{prob}_{\mathbf{I},o}(\phi_{ig})$	$\mathrm{prob}_{\mathbf{I},o}(\psi_{in})$	$\mathrm{prob}_{\mathbf{I},o}(\psi_{ig})$
o_1	undefined	undefined	undefined	undefined
o_2	$[0.40, 0.60]$	$[0.40, 0.60]$	$[0.33, 0.49]$	$[0.22, 0.60]$
o_3	$[1.00, 1.00]$	$[1.00, 1.00]$	$[0.00, 0.00]$	$[0.00, 0.00]$
o_4	$[0.00, 0.00]$	$[0.00, 0.00]$	$[0.67, 0.67]$	$[0.67, 0.67]$
o_5	$[0.40, 0.60]$	$[0.40, 0.60]$	$[0.27, 0.40]$	$[0.07, 0.60]$
o_6	$[1.00, 1.00]$	$[1.00, 1.00]$	$[0.00, 0.00]$	$[0.00, 0.00]$
o_7	$[1.00, 1.00]$	$[1.00, 1.00]$	$[0.00, 0.00]$	$[0.00, 0.00]$

We are now ready to assign truth values to probabilistic selection conditions:

Definition 7.7 (satisfaction of probabilistic selection conditions) Let $\mathbf{I} = (\pi, \nu)$ be a POB-instance over the POB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ and let $o \in \pi(\mathcal{C})$. We extend $\mathrm{prob}_{\mathbf{I},o}$ to probabilistic selection conditions as follows:

- $\mathrm{prob}_{\mathbf{I},o} \models (\phi)[L, U]$ iff $\mathrm{prob}_{\mathbf{I},o}(\phi) \subseteq [L, U]$.
- $\mathrm{prob}_{\mathbf{I},o} \models \neg\phi$ iff it is not the case that $\mathrm{prob}_{\mathbf{I},o} \models \phi$.
- $\mathrm{prob}_{\mathbf{I},o} \models \phi \wedge \psi$ iff $\mathrm{prob}_{\mathbf{I},o} \models \phi$ and $\mathrm{prob}_{\mathbf{I},o} \models \psi$.

- $\text{prob}_{\mathbf{I},o} \models \phi \vee \psi$ iff $\text{prob}_{\mathbf{I},o} \models \phi$ or $\text{prob}_{\mathbf{I},o} \models \psi$.

Let us give an illustrating example.

Example 7.6 (Plant Example: satisfaction of probabilistic selection conditions) In the Plant Example, it is easy to see that:

- $\text{prob}_{\mathbf{I},o_2} \models (x.\mathsf{sun} = \mathsf{mild} \otimes_{in} x.\mathsf{rain} > 21)[0.3, 0.5]$ (see Example 7.5).
- $\text{prob}_{\mathbf{I},o_2} \not\models (x.\mathsf{sun} = \mathsf{mild} \otimes_{ig} x.\mathsf{rain} > 21)[0.3, 0.5]$ (see Example 7.5).
- $\text{prob}_{\mathbf{I},o_2} \models (x.\mathsf{sun} = \mathsf{mild})[0.4, 1] \wedge (x.\mathsf{rain} > 21)[0.8, 1]$ (see Example 7.4).
- $\text{prob}_{\mathbf{I},o_3} \not\models (x.\mathsf{sun} = \mathsf{mild})[0.4, 1] \wedge (x.\mathsf{rain} > 21)[0.8, 1]$ (see Example 7.4).

After these preparations, we are finally ready to define the selection operation.

Definition 7.8 (selection on POB-instances) Let $\mathbf{I} = (\pi, \nu)$ be a POB-instance over the POB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \text{me}, \wp)$ and let ϕ be a probabilistic selection condition over the object variable x. The *selection on* $\mathbf{I}$ *with respect to* ϕ, denoted $\sigma_\phi(\mathbf{I})$, is the POB-instance (π', ν') over $\mathbf{S}$, where:

- $\pi'(c) = \{o \in \pi(c) \mid \text{prob}_{\mathbf{I},o} \models \phi\}$.
- $\nu' = \nu \mid \pi'(\mathcal{C})$ (that is, the mapping ν restricted to $\pi'(\mathcal{C})$).

The following example shows precisely what happens in the case of the Plant Example when we perform selection with respect to probabilistic selection conditions.

Example 7.7 (Plant Example: selection) In the Plant Example, the selection on $\mathbf{I} = (\pi, \nu)$ with respect to the probabilistic selection condition $(x.\mathsf{sun} = \mathsf{mild})[0.4, 1] \wedge (x.\mathsf{rain} > 21)[0.8, 1]$ is the POB-instance (π', ν') over $\mathbf{S}$ (see Example 7.4), where π' and ν' are shown in Table 5. This result

c	$\pi'(c)$
plants	$\{\}$
annuals	$\{\}$
perennials	$\{\}$
vegetables	$\{\}$
herbs	$\{\}$
flowers	$\{\}$
annuals_herbs	$\{\}$
perennials_flowers	$\{o_4\}$

oid	$\nu'(oid)$
o_2	$[\mathsf{pname}: \langle\{\mathsf{Cuban\text{-}Basil}, \mathsf{Lemon\text{-}Basil}\}, \mathrm{u}, \mathrm{u}\rangle$, $\mathsf{soil}: \langle\{\mathsf{loamy}, \mathsf{sandy}\}, 0.7\,\mathrm{u}, 1.3\,\mathrm{u}\rangle$, $\mathsf{rain}: \langle\{20, \ldots, 30\}, \mathrm{u}, \mathrm{u}\rangle$, $\mathsf{sun}: \langle\{\mathsf{mild}, \mathsf{medium}\}, 0.8\,\mathrm{u}, 1.2\,\mathrm{u}\rangle$, $\mathsf{expyears}: \langle\{2, 3, 4\}, 0.6\,\mathrm{u}, 1.8\,\mathrm{u}\rangle$, $\mathsf{classification}: \langle\{\mathsf{french}, \mathsf{silver}, \mathsf{wooly}\}, 0.6\,\mathrm{u}, 1.8\,\mathrm{u}\rangle]$

Table 5: π' and ν' resulting from selection

is also obtained by the selection on $\mathbf{I}$ with respect to $(x.\mathsf{sun} = \mathsf{mild} \otimes_{in} x.\mathsf{rain} > 21)[0.3, 0.5]$ (see Example 7.5). The selection on $\mathbf{I}$ with respect to $(x.\mathsf{sun} = \mathsf{mild} \otimes_{ig} x.\mathsf{rain} > 21)[0.3, 0.5]$, in contrast, produces the empty POB-instance over $\mathbf{S}$ (see Example 7.5).

7.2 Projection

In this section, we define projection of POB-instances on arbitrary sets of attributes. We first define the projection of POB-schemas on sets of attributes.

Definition 7.9 (projection of POB-schemas) Let $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ be a POB-schema and let $\mathbf{A}$ be a set of attributes. The *projection of* $\mathbf{S}$ *on* $\mathbf{A}$, denoted $\Pi_{\mathbf{A}}(\mathbf{S})$, is the POB-schema $(\mathcal{C}, \sigma', \Rightarrow, \mathrm{me}, \wp)$, where the new type $\sigma'(c)$ of each class $c \in \mathcal{C}$ is obtained from the old type $\sigma(c) = [B_1\colon \tau_1, \ldots, B_k\colon \tau_k]$ by deleting all $B_j\colon \tau_j$'s with $B_j \notin \mathbf{A}$.

Let us consider an example to illustrate the projection of POB-schemas.

Example 7.8 (Plant Example: projection of POB-schemas) Let the POB-schema $\mathbf{S}$ be given by the POB-schema described in Example 5.4. Then, the projection of $\mathbf{S}$ on the set of attributes $\mathbf{A} = \{\mathsf{pname}, \mathsf{rain}\}$ has the type assignment σ' shown in Table 6.

c	$\sigma'(c)$
plants	[pname: string, rain : integer]
annuals	[pname: string, rain : integer]
perennials	[pname: string, rain : integer]
vegetables	[pname: string, rain : integer]
herbs	[pname: string, rain : integer]
flowers	[pname: string, rain : integer]
annuals_herbs	[pname: string, rain : integer]
perennials_flowers	[pname: string, rain : integer]

Table 6: Type assignment σ' resulting from schema projection

Given a consistent POB-schema as input, the projection operation always produces a consistent POB-schema as output. This is shown by the following theorem.

Theorem 7.1 *Let* $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ *be a POB-schema and let* $\mathbf{A}$ *be a set of attributes. If* $\mathbf{S}$ *is consistent, then* $\Pi_{\mathbf{A}}(\mathbf{S})$ *is consistent.*

We next define the projection of probabilistic tuple values.

Definition 7.10 (projection of probabilistic tuple values) Let ptv be a probabilistic tuple value of the form $[B_1\colon (V_1, \alpha_1, \beta_1), \ldots, B_k\colon (V_k, \alpha_k, \beta_k)]$ and let $\mathbf{A}$ be a set of attributes. The *projection* of ptv on $\mathbf{A}$, denoted $\Pi_{\mathbf{A}}(ptv)$, is obtained from $[B_1\colon (V_1, \alpha_1, \beta_1), \ldots, B_k\colon (V_k, \alpha_k, \beta_k)]$ by deleting all $B_j\colon (V_j, \alpha_j, \beta_j)$'s with $B_j \notin \mathbf{A}$.

We give a small example to illustrate projection of probabilistic tuple values.

Example 7.9 (Plant Example: projection of probabilistic tuple values) Let the probabilistic tuple value ptv be given as follows (note that ptv is associated with the object o_2 in Example 6.2):

$$ptv = [\ \mathsf{pname}\colon \langle\{\mathsf{Cuban\text{-}Basil}, \mathsf{Lemon\text{-}Basil}\}, \mathrm{u}, \mathrm{u}\rangle,$$

$\mathsf{soil}\colon \langle\{\mathsf{loamy}, \mathsf{sandy}\}, 0.7\,\mathrm{u}, 1.3\,\mathrm{u}\rangle,$
$\mathsf{rain}\colon \langle\{20, \ldots, 30\}, \mathrm{u}, \mathrm{u}\rangle,$
$\mathsf{sun}\colon \langle\{\mathsf{mild}, \mathsf{medium}\}, 0.8\,\mathrm{u}, 1.2\,\mathrm{u}\rangle,$
$\mathsf{expyears}\colon \langle\{2, 3, 4\}, 0.6\,\mathrm{u}, 1.8\,\mathrm{u}\rangle,$
$\mathsf{classification}\colon \langle\{\mathsf{french}, \mathsf{silver}, \mathsf{wooly}\}, 0.6\,\mathrm{u}, 1.8\,\mathrm{u}\rangle]$

The projection of ptv on the set of attributes $\mathbf{A} = \{\mathsf{pname}, \mathsf{rain}\}$ is given as follows:

$$\Pi_{\mathbf{A}}(ptv) \;=\; [\mathsf{pname}\colon \langle\ \mathsf{Cuban\text{-}Basil}, \mathsf{Lemon\text{-}Basil}\}, \mathrm{u}, \mathrm{u}\rangle, \mathsf{rain}\colon \langle\{20, \ldots, 30\}, \mathrm{u}, \mathrm{u}\rangle]$$

We are now ready to define the projection of POB-instances.

Definition 7.11 (projection of POB-instances) Let $\mathbf{I} = (\pi, \nu)$ be a POB-instance over the POB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ and let $\mathbf{A}$ be a set of attributes. The *projection of* $\mathbf{I}$ *on* $\mathbf{A}$, denoted $\Pi_{\mathbf{A}}(\mathbf{I})$, is defined as the POB-instance (π', ν') over the POB-schema $\Pi_{\mathbf{A}}(\mathbf{S})$, where:

- $\pi'(c) = \pi(c)$ for all classes $c \in \mathcal{C}$.
- $\nu'(o) = \Pi_{\mathbf{A}}(\nu(o))$ for all oids $o \in \pi(\mathcal{C})$.

Let us illustrate this definition within the Plant Example.

Example 7.10 (Plant Example: projection of POB-instances) Let the POB-instance $\mathbf{I} = (\pi, \nu)$ be given by the POB-instance described in Example 6.2. The projection of $\mathbf{I}$ on $\mathbf{A} = \{\mathsf{pname}, \mathsf{rain}\}$ is the POB-instance (π', ν'), where π' and ν' are given Table 7 (note that π' is the same as π).

c	$\pi'(c)$
plants	$\{o_1\}$
annuals	$\{\}$
perennials	$\{\}$
vegetables	$\{\}$
herbs	$\{\}$
flowers	$\{\}$
annuals_herbs	$\{o_2, o_3, o_5, o_6, o_7\}$
perennials_flowers	$\{o_4\}$

oid	$\nu'(oid)$
o_1	$[\mathsf{pname}\colon \langle\{\mathsf{Lady\text{-}Fern}, \mathsf{Ostrich\text{-}Fern}\}, \mathrm{u}, \mathrm{u}\rangle,$ $\mathsf{rain}\colon \langle\{25, \ldots, 30\}, \mathrm{u}, \mathrm{u}\rangle]$
o_2	$[\mathsf{pname}\colon \langle\{\mathsf{Cuban\text{-}Basil}, \mathsf{Lemon\text{-}Basil}\}, \mathrm{u}, \mathrm{u}\rangle,$ $\mathsf{rain}\colon \langle\{20, \ldots, 30\}, \mathrm{u}, \mathrm{u}\rangle]$
o_3	$[\mathsf{pname}\colon \langle\{\mathsf{Mint}\}, \mathrm{u}, \mathrm{u}\rangle, \mathsf{rain}\colon \langle\{20\}, \mathrm{u}, \mathrm{u}\rangle]$
o_4	$[\mathsf{pname}\colon \langle\{\mathsf{Aster}, \mathsf{Salvia}\}, \mathrm{u}, \mathrm{u}\rangle,$ $\mathsf{rain}\colon \langle\{20, \ldots, 25\}, \mathrm{u}, \mathrm{u}\rangle]$
o_5	$[\mathsf{pname}\colon \langle\{\mathsf{Thyme}\}, \mathrm{u}, \mathrm{u}\rangle, \mathsf{rain}\colon \langle\{20, \ldots, 25\}, \mathrm{u}, \mathrm{u}\rangle]$
o_6	$[\mathsf{pname}\colon \langle\{\mathsf{Mint}\}, \mathrm{u}, \mathrm{u}\rangle, \mathsf{rain}\colon \langle\{20\}, \mathrm{u}, \mathrm{u}\rangle]$
o_7	$[\mathsf{pname}\colon \langle\{\mathsf{Sage}\}, \mathrm{u}, \mathrm{u}\rangle, \mathsf{rain}\colon \langle\{20, 21\}, \mathrm{u}, \mathrm{u}\rangle]$

Table 7: π' and ν' resulting from projection

7.3 Renaming

In this section, we define renaming of (top-level) attributes in POB-instances. This operation is especially useful in connection with cartesian product and join (see Sections 7.4 and 7.5). We first define renaming conditions.

Definition 7.12 (renaming condition) Let $\mathrm{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ be a POB-schema and let $\mathbf{A}$ be the set of all top-level attribute names of S. A *renaming condition* is an expression of the form $\vec{\mathbf{B}} \leftarrow \vec{\mathbf{C}}$, where $\vec{\mathbf{B}} = B_1, B_2, \ldots, B_l$ is a list of distinct attribute names from $\mathbf{A}$, and $\vec{\mathbf{C}} = C_1, C_2, \ldots, C_l$ is a list of distinct attributes from $\mathcal{A} - (\mathbf{A} - \{B_1, B_2, \ldots, B_l\})$ (this condition ensures that each attribute C_i that is in $\mathbf{A}$ must also occur in $\mathbf{B}$, i.e., be renamed itself).

Let us give an example of a renaming condition within the Plant Example.

Example 7.11 (Plant Example: renaming condition) Let us consider the POB-schema computed in Example 7.8. A renaming condition is given by $\mathsf{pname}, \mathsf{rain} \leftarrow \mathsf{pname2}, \mathsf{rain2}$.

We now define the renaming of attributes in POB-schemas.

Definition 7.13 (renaming in POB-schemas) Let $\mathrm{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ be a POB-schema and let $N = B_1, B_2, \ldots, B_l \leftarrow C_1, C_2, \ldots, C_l$ be a renaming condition. The *renaming in* S *with respect to* N, denoted $\delta_N(\mathrm{S})$, is the POB-schema $(\mathcal{C}, \sigma', \Rightarrow, \mathrm{me}, \wp)$, where the new type $\sigma'(c)$ of each class $c \in \mathcal{C}$ is obtained from the old type $\sigma(c) = [A_1\colon \tau_1, \ldots, A_k\colon \tau_k]$ by replacing each attribute A_j with $A_j = B_i$ for some $i \in \{1, \ldots, l\}$ by the new attribute C_i.

Note. Though the above definition does not include renaming of nested attributes, this may be accomplished by a straightforward extension. For the sake of simplicity, we skip this.

Let us give an example to illustrate the renaming of attributes in POB-schemas.

Example 7.12 (Plant Example: renaming of POB-schemas) Let the POB-schema S be given by the POB-schema computed in Example 7.8. The renaming of S with respect to the renaming condition $\mathsf{pname}, \mathsf{rain} \leftarrow \mathsf{pname2}, \mathsf{rain2}$ has the following type assignment σ':

c	$\sigma'(c)$
plants	[pname2: string, rain2: integer]
annuals	[pname2: string, rain2: integer]
perennials	[pname2: string, rain2: integer]
vegetables	[pname2: string, rain2: integer]
herbs	[pname2: string, rain2: integer]
flowers	[pname2: string, rain2: integer]
annuals_herbs	[pname2: string, rain2: integer]
perennials_flowers	[pname2: string, rain2: integer]

Given a consistent POB-schema as input, the renaming operation always produces a consistent POB-schema as output. This is shown by the following theorem.

Theorem 7.2 *Let* $\mathrm{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ *be a POB-schema and let* N *be a renaming condition. If* S *is consistent, then* $\delta_N(\mathrm{S})$ *is consistent.*

We next define the renaming of attributes in probabilistic tuple values.

Definition 7.14 (renaming of probabilistic tuple values) Let ptv be a probabilistic tuple value of the form $[A_1 \colon (V_1, \alpha_1, \beta_1), \ldots, A_k \colon (V_k, \alpha_k, \beta_k)]$ and let $N = B_1, B_2, \ldots, B_l \leftarrow C_1, C_2, \ldots, C_l$ be a renaming condition. The *renaming of ptv with respect to N*, denoted $\delta_N(ptv)$, is obtained from $[A_1 \colon (V_1, \alpha_1, \beta_1), \ldots, A_k \colon (V_k, \alpha_k, \beta_k)]$ by replacing each attribute A_j with $A_j = B_i$ for some $i \in \{1, \ldots, l\}$ by the new attribute C_i.

We give a small example to illustrate the renaming of attributes in probabilistic tuple values.

Example 7.13 (Plant Example: renaming in probabilistic tuple values) Let the probabilistic tuple value ptv taken from Example 7.9 be as follows:

$$ptv = [\mathsf{pname} \colon \langle \{\mathsf{Cuban\text{-}Basil}, \mathsf{Lemon\text{-}Basil}\}, \mathrm{u}, \mathrm{u} \rangle, \mathsf{rain} \colon \langle \{20, \ldots, 30\}, \mathrm{u}, \mathrm{u} \rangle]$$

The renaming in ptv with respect to pname, rain $\leftarrow$ pname2, rain2 is given as follows:

$$ptv = [\mathsf{pname2} \colon \langle \{\mathsf{Cuban\text{-}Basil}, \mathsf{Lemon\text{-}Basil}\}, \mathrm{u}, \mathrm{u} \rangle, \mathsf{rain2} \colon \langle \{20, \ldots, 30\}, \mathrm{u}, \mathrm{u} \rangle]$$

We are now ready to define the renaming of attributes in POB-instances.

Definition 7.15 (renaming in POB-instances) Let $\mathbf{I} = (\pi, \nu)$ be a POB-instance over the POB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ and let N be a renaming condition. The *renaming in $\mathbf{I}$ with respect to N*, denoted $\delta_N(\mathbf{I})$, is defined as the POB-instance (π', ν') over the POB-schema $\delta_N(\mathbf{S})$, where:

- $\pi'(c) = \pi(c)$ for all classes $c \in \mathcal{C}$.
- $\nu'(o) = \delta_N(\nu(o))$ for all oids $o \in \pi(\mathcal{C})$.

Let us illustrate this definition within the Plant Example.

Example 7.14 (Plant Example: renaming in POB-instances) Let the POB-instance $\mathbf{I} = (\pi, \nu)$ be given by the POB-instance computed in Example 7.10. The renaming in $\mathbf{I}$ with respect to the renaming condition pname, rain $\leftarrow$ pname2, rain2 is the POB-instance (π', ν'), where π' and ν' are given in Table 8 (note that π' is the same as π).

7.4 Cartesian Product

In this section, we define the cartesian product of two POB-instances. In classical relational databases, the cartesian product of two relations consists of the set of all tuples that can be obtained by concatenating a tuple in the first relation with a tuple in the second relation. If one follows this intuition, the cartesian product of two POB-instances should be obtained by concatenating the property list of any object in the first POB-instance with the property list of any object in the second POB-instance. This will be the intuition underlying our definition of cartesian product.

Let us first come back to the Plant Example to show that the cartesian product is meaningful.

c	$\pi'(c)$
plants	$\{o_1\}$
annuals	$\{\}$
perennials	$\{\}$
vegetables	$\{\}$
herbs	$\{\}$
flowers	$\{\}$
annuals_herbs	$\{o_2, o_3, o_5, o_6, o_7\}$
perennials_flowers	$\{o_4\}$

oid	$\nu'(oid)$
o_1	$[\mathsf{pname2}\colon \langle\{\mathsf{Lady\text{-}Fern}, \mathsf{Ostrich\text{-}Fern}\}, \mathrm{u}, \mathrm{u}\rangle,$ $\mathsf{rain2}\colon \langle\{25, \ldots, 30\}, \mathrm{u}, \mathrm{u}\rangle]$
o_2	$[\mathsf{pname2}\colon \langle\{\mathsf{Cuban\text{-}Basil}, \mathsf{Lemon\text{-}Basil}\}, \mathrm{u}, \mathrm{u}\rangle,$ $\mathsf{rain2}\colon \langle\{20, \ldots, 30\}, \mathrm{u}, \mathrm{u}\rangle]$
o_3	$[\mathsf{pname2}\colon \langle\{\mathsf{Mint}\}, \mathrm{u}, \mathrm{u}\rangle,$ $\mathsf{rain2}\colon \langle\{20\}, \mathrm{u}, \mathrm{u}\rangle]$
o_4	$[\mathsf{pname2}\colon \langle\{\mathsf{Aster}, \mathsf{Salvia}\}, \mathrm{u}, \mathrm{u}\rangle,$ $\mathsf{rain2}\colon \langle\{20, \ldots, 25\}, \mathrm{u}, \mathrm{u}\rangle]$
o_5	$[\mathsf{pname2}\colon \langle\{\mathsf{Thyme}\}, \mathrm{u}, \mathrm{u}\rangle,$ $\mathsf{rain2}\colon \langle\{20, \ldots, 25\}, \mathrm{u}, \mathrm{u}\rangle]$
o_6	$[\mathsf{pname2}\colon \langle\{\mathsf{Mint}\}, \mathrm{u}, \mathrm{u}\rangle,$ $\mathsf{rain2}\colon \langle\{20\}, \mathrm{u}, \mathrm{u}\rangle]$
o_7	$[\mathsf{pname2}\colon \langle\{\mathsf{Sage}\}, \mathrm{u}, \mathrm{u}\rangle,$ $\mathsf{rain2}\colon \langle\{20, 21\}, \mathrm{u}, \mathrm{u}\rangle]$

Table 8: π' and ν' resulting from renaming

Example 7.15 (Plant Example: cartesian product) Suppose we are interested in pairs of plants that flourish with a certain probability in the same environment (for example, in pairs of plants that have the same rain requirements with some probability). To obtain this information, we must somehow connect the knowledge tied to each oid with the knowledge tied to other oids.

The first challenge in defining the cartesian product of two POB-instances is the following. Suppose we know that the POB-schemas of our two POB-instances are $\mathbf{S}_1 = (\mathcal{C}_1, \sigma_1, \Rightarrow_1, \mathrm{me}_1, \wp_1)$ and $\mathbf{S}_2 = (\mathcal{C}_2, \sigma_2, \Rightarrow_2, \mathrm{me}_2, \wp_2)$. Let $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ denote the schema of the cartesian product instance. What should the relationship between $\mathbf{S}_1, \mathbf{S}_2$, and $\mathbf{S}$ be?

Recall that in classical relational databases, the cartesian product $R_1 \times R_2$ of relation schemes R_1 and R_2 is only defined if they have disjoint sets of attributes. Further, $R_1 \times R_2$ and $R_2 \times R_1$ yield the same schema; similarly, we desire that $\mathbf{S}_1 \times \mathbf{S}_2 = \mathbf{S}_2 \times \mathbf{S}_1$ holds. This needs special care.

In the rest of this paper, we assume that for each scheme $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$, $\mathcal{C}$ is a database relation, i.e., the classes $c \in \mathcal{C}$ are tuples over a relation scheme $R = R(\mathbf{S})$ associated with $\mathbf{S}$. Two POB-schemas $\mathbf{S}_1$ and $\mathbf{S}_2$ can be combined using cartesian product under the following condition:

Definition 7.16 (cartesian-product-compatible POB-schemas) The two POB-schemas $\mathbf{S}_1 = (\mathcal{C}_1, \sigma_1, \Rightarrow_1, \mathrm{me}_1, \wp_1)$ and $\mathbf{S}_2 = (\mathcal{C}_2, \sigma_2, \Rightarrow_2, \mathrm{me}_2, \wp_2)$ are *cartesian-product-compatible* iff $R(\mathbf{S}_1)$ and $R(\mathbf{S}_2)$ are disjoint and for all classes $c_1 \in \mathcal{C}_1$ and $c_2 \in \mathcal{C}_2$: $\sigma_1(c_1)$ and $\sigma_2(c_2)$ are tuple types over disjoint sets of attributes.

Note that any POB-schemas $\mathbf{S}_1$ and $\mathbf{S}_2$ can be made cartesian product compatible by renaming of attributes in $R(\mathbf{S}_1)$, $R(\mathbf{S}_2)$ and attributes of tuple types.

Definition 7.17 (cartesian product of POB-schemas) Let $\mathbf{S}_1 = (\mathcal{C}_1, \sigma_1, \Rightarrow_1, \mathrm{me}_1, \wp_1)$ and $\mathbf{S}_2 = (\mathcal{C}_2, \sigma_2, \Rightarrow_2, \mathrm{me}_2, \wp_2)$ be two cartesian-product-compatible POB-schemas, and let $R_1 = R(\mathbf{S}_1)$

and $R_2 = R(\mathbf{S}_2)$. The *cartesian product* of $\mathbf{S}_1$ and $\mathbf{S}_2$, denoted $\mathbf{S}_1 \times \mathbf{S}_2$, is the POB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ such that:

- $\mathcal{C} = \mathcal{C}_1 \times \mathcal{C}_2$.
- For all $c \in \mathcal{C}$, let $\sigma(c[R_1], c[R_2]) = [A_1 \colon \tau_1, \ldots, A_k \colon \tau_k, A_{k+1} \colon \tau_{k+1}, \ldots, A_{k+m} \colon \tau_{k+m}]$, where $\sigma_1(c[R_1]) = [A_1 \colon \tau_1, \ldots, A_k \colon \tau_k]$ and $\sigma_2(c[R_2]) = [A_{k+1} \colon \tau_{k+1}, \ldots, A_{k+m} \colon \tau_{k+m}]$.[4]
- The directed acyclic graph $(\mathcal{C}, \Rightarrow)$ is defined as follows. For all $c, d \in \mathcal{C}$:

$$c \Rightarrow d \quad \text{iff} \quad (c[R_1] \Rightarrow_1 d[R_1] \wedge c[R_2] = d[R_2]) \text{ or } (c[R_1] = d[R_1] \wedge c[R_2] \Rightarrow_2 d[R_2]).$$

- The partitioning me is given as follows. For all $c \in \mathcal{C}$:

$$\mathrm{me}(c) \;=\; \{\mathcal{P}_1 \times \{c[R_2]\} \mid \mathcal{P}_1 \in \mathrm{me}_1(c[R_1])\} \cup \{\{c[R_1]\} \times \mathcal{P}_2 \mid \mathcal{P}_2 \in \mathrm{me}_2(c[R_2])\}.$$

- The probability assignment $\wp$ is defined as follows. For all $c \Rightarrow d$:

$$\wp(c, d) \;=\; \begin{cases} \wp_1(c[R_1], d[R_1]) & \text{if } c[R_2] = d[R_2] \\ \wp_2(c[R_2], d[R_2]) & \text{if } c[R_1] = d[R_1]\,. \end{cases}$$

(Note that $\mathcal{C} = \mathcal{C}_1 \times \mathcal{C}_2$ implicitly defines that $R(\mathbf{S}) = R_1 \cup R_2$.)

Let us illustrate this definition within the Plant Example.

Example 7.16 (Plant Example: cartesian product of POB-schemas) Let $\mathbf{S}_1$ be the POB-schema computed in Example 7.8, and let $\mathbf{S}_2$ be the POB-schema computed in Example 7.12 in which each class c is replaced by c'. The cartesian product schema $\mathbf{S}_1 \times \mathbf{S}_2 = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ is as follows:

- A partial view on the set of classes $\mathcal{C}$ is given in Figure 4 (note that we use pl, an, pe, ve, he, fl, ah, and pf as abbreviations for plants, annuals, perennials, vegetables, herbs, flowers, annuals_herbs, and perennials_flowers, respectively).
- Each class $c \in \mathcal{C}$ is assigned the following type under σ:

$$\sigma(c) = [\mathsf{pname}\colon \mathsf{string}, \mathsf{rain}\colon \mathsf{integer}, \mathsf{pname2}\colon \mathsf{string}, \mathsf{rain2}\colon \mathsf{integer}]\,.$$

- A partial view on the directed acyclic graph $(\mathcal{C}, \Rightarrow)$, the partitioning me, and the probability assignment $\wp$ is also given in Figure 4.

The cartesian product of two consistent POB-schemas is always consistent:

[4] As usual, $c[U]$ denotes the restriction of tuple c to the attributes in U.

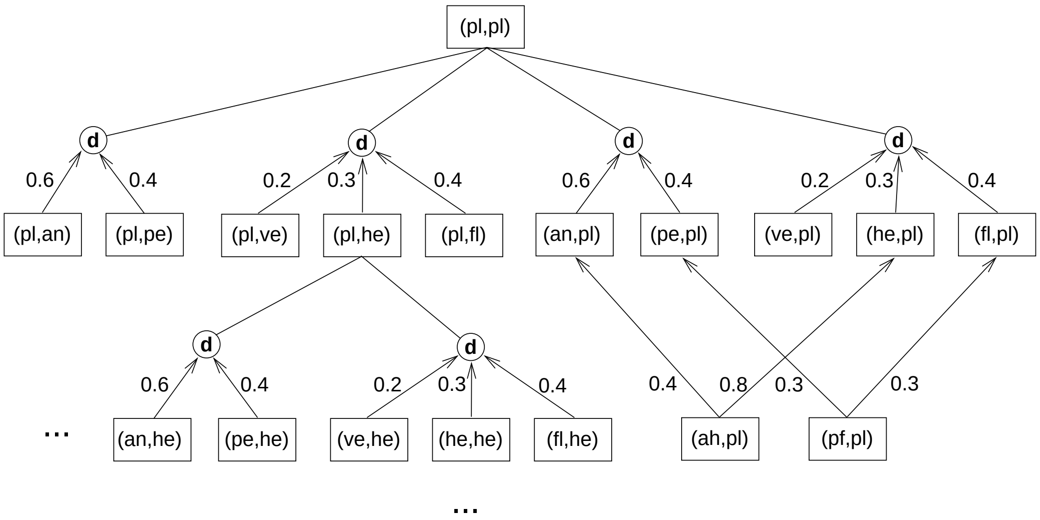


Figure 4: Some classes in the cartesian product of the Plant Example

Theorem 7.3 *Let* S_1 *and* S_2 *be two cartesian-product-compatible POB-schemas. If* S_1 *and* S_2 *are consistent, then* $\mathrm{S}_1 \times \mathrm{S}_2$ *is consistent.*

Proof. Let $\mathrm{S}_1 = (\mathcal{C}_1, \sigma_1, \Rightarrow_1, \mathrm{me}_1, \wp_1)$, $\mathrm{S}_2 = (\mathcal{C}_2, \sigma_2, \Rightarrow_2, \mathrm{me}_2, \wp_2)$, and $\mathrm{S}_1 \times \mathrm{S}_2 = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$. Let $\varepsilon_1 : \mathcal{C}_1 \to 2^{\mathcal{O}_1}$ and $\varepsilon_2 : \mathcal{C}_2 \to 2^{\mathcal{O}_2}$ be models of S_1 and S_2, respectively. Let the mapping $\varepsilon : \mathcal{C} \to 2^{\mathcal{O}}$, where $\mathcal{C} = \mathcal{C}_1 \times \mathcal{C}_2$ and $\mathcal{O} = \mathcal{O}_1 \times \mathcal{O}_2$, be defined as follows:

$$\varepsilon(c) \;=\; \varepsilon_1(c[R_1]) \times \varepsilon_2(c[R_2]), \;\; \text{for all } c \in \mathcal{C}\,.$$

We now show that ε is a model of S. We first prove C1. Since $\varepsilon_1(c_1) \neq \emptyset$ for all classes $c_1 \in \mathcal{C}_1$ and $\varepsilon_2(c_2) \neq \emptyset$ for all classes $c[R_2] \in \mathcal{C}_2$, we get $\varepsilon(c) \neq \emptyset$ for all classes $c \in \mathcal{C}$. We next show C2 and C4. Let $c, d \in \mathcal{C}$ with $c \Rightarrow d$. Without loss of generality, we can assume that $c[R_1] \Rightarrow_1 d[R_1]$ and $c[R_2] = d[R_2]$. Since ε_1 is a model of S_1, it holds that $\varepsilon_1(c[R_1]) \subseteq \varepsilon_1(d[R_1])$ and $|\varepsilon_1(c[R_1])| = \wp_1(c[R_1], d[R_1]) \cdot |\varepsilon_1(d[R_1])|$. Hence, it immediately follows $\varepsilon(c) \subseteq \varepsilon(d)$ and $|\varepsilon(c)| = \wp(c, d) \cdot |\varepsilon(d)|$. We finally prove C3. Let $c, d \in \mathcal{C}$ be two distinct classes that belong to the same cluster $\mathcal{P} \in \bigcup \mathrm{me}(\mathcal{C})$. Without loss of generality, we can assume that $c[R_1], d[R_1] \in \mathcal{C}_1$ belong to the same cluster $\mathcal{P}_1 \in \bigcup \mathrm{me}_1(\mathcal{C}_1)$ and that $c[R_2] = d[R_2]$. Since ε_1 is a model of S_1, it holds that $\varepsilon_1(c[R_1]) \cap \varepsilon_1(d[R_1]) = \emptyset$. Thus, it follows that $\varepsilon(c) \cap \varepsilon(d) = \emptyset$. $\square$

We now define the cartesian product of probabilistic tuple values.

Definition 7.18 (cartesian product of probabilistic tuple values) Let ptv_1 and ptv_2 be two probabilistic tuple values over the disjoint sets of attributes $\mathbf{A}_1$ and $\mathbf{A}_2$, respectively. The *cartesian product* of ptv_1 and ptv_2, denoted $ptv_1 \times ptv_2$, is the probabilistic tuple value ptv over the set of attributes $\mathbf{A}_1 \cup \mathbf{A}_2$ defined by:

- $ptv.A = ptv_1.A$ for all attributes $A \in \mathbf{A}_1$.

- $ptv.A = ptv_2.A$ for all attributes $A \in \mathbf{A}_2$.

Note that $ptv_1 \times ptv_2 = ptv_2 \times ptv_1$, since by convention the ordering of attributes in a complex value tuple is immaterial.

Example 7.17 (Plant Example: cartesian product of probabilistic tuple values) Let us consider the following two probabilistic tuple values (taken from Examples 7.10 and 7.15, respectively):

$$
\begin{aligned}
ptv_1 &= [\mathsf{pname}\colon \langle\{\mathsf{Cuban\text{-}Basil}, \mathsf{Lemon\text{-}Basil}\}, \mathrm{u}, \mathrm{u}\rangle, \mathsf{rain}\colon \langle\{20, \ldots, 30\}, \mathrm{u}, \mathrm{u}\rangle] \\
ptv_2 &= [\mathsf{pname2}\colon \langle\{\mathsf{Mint}\}, \mathrm{u}, \mathrm{u}\rangle, \mathsf{rain2}\colon \langle\{20\}, \mathrm{u}, \mathrm{u}\rangle]\,.
\end{aligned}
$$

The cartesian product of ptv_1 and ptv_2 is given as follows:

$$
\begin{aligned}
ptv_1 \times ptv_2 = [&\mathsf{pname}\colon \langle\{\mathsf{Cuban\text{-}Basil}, \mathsf{Lemon\text{-}Basil}\}, \mathrm{u}, \mathrm{u}\rangle, \mathsf{rain}\colon \langle\{20, \ldots, 30\}, \mathrm{u}, \mathrm{u}\rangle, \\
&\mathsf{pname2}\colon \langle\{\mathsf{Mint}\}, \mathrm{u}, \mathrm{u}\rangle, \mathsf{rain2}\colon \langle\{20\}, \mathrm{u}, \mathrm{u}\rangle].
\end{aligned}
$$

We are finally ready to define the cartesian product of two POB-instances. As in the case of classes, we assume in the rest of this paper that each oid $o \in \mathcal{O}$ that occurs in a POB-instance $\mathbf{I} = (\pi, \nu)$ over $\mathbf{S}$ is a tuple for the relation scheme $R(\mathbf{S})$; each such o may be written as the list $(o_1, \ldots, o_m)$ of the values for the attributes $A_1, \ldots, A_m$ of $R(\mathbf{S})$.

Definition 7.19 (cartesian product of POB-instances) Let $\mathbf{I}_1 = (\pi_1, \nu_1)$ and $\mathbf{I}_2 = (\pi_2, \nu_2)$ be two POB-instances over the cartesian-product-compatible POB-schemas $\mathbf{S}_1 = (\mathcal{C}_1, \sigma_1, \Rightarrow_1, \mathrm{me}_1, \wp_1)$ and $\mathbf{S}_2 = (\mathcal{C}_2, \sigma_2, \Rightarrow_2, \mathrm{me}_2, \wp_2)$, respectively. The *cartesian product* of $\mathbf{I}_1$ and $\mathbf{I}_2$, denoted $\mathbf{I}_1 \times \mathbf{I}_2$, is defined as the POB-instance (π, ν) over the POB-schema $\mathbf{S} = \mathbf{S}_1 \times \mathbf{S}_2$, where

- $\pi(c) = \pi_1(c[R_1]) \times \pi_2(c[R_2])$, for all $c \in \mathcal{C}$ (here, $\pi_1(c[R_1]) \times \pi_2(c[R_2]) \subseteq \mathcal{O}$ is assumed).
- $\nu(o) = \nu_1(o[R_1]) \times \nu_2(o[R_2])$, for all $o \in \pi(\mathcal{C})$.

Let us illustrate this definition within the Plant Example.

Example 7.18 (Plant Example: cartesian product of POB-instances) Let $\mathbf{I}_1$ and $\mathbf{I}_2$ be the POB-instances computed in Examples 7.10 and 7.15, respectively. The cartesian product of $\mathbf{I}_1$ and $\mathbf{I}_2$ is the POB-instance (π, ν), where partial views of π and ν are given in Table 9:

7.5 Join

In classical relational databases, the join operator is a generalization of the cartesian product. This will also be the case for the join of POB-instances, which is defined in this section. We start with the notion of join-compatibility.

c	$\pi(c)$
$(\mathsf{pl}, \mathsf{pl})$	$\{(o_1, o_1)\}$
$(\mathsf{an}, \mathsf{pl})$	$\{\}$
$(\mathsf{ah}, \mathsf{pl})$	$\{(o_2, o_1), (o_3, o_1),$ $(o_5, o_1), (o_6, o_1), (o_7, o_1)\}$
$(\mathsf{pf}, \mathsf{pl})$	$\{(o_4, o_1)\}$

oid	$\nu(oid)$
(o_1, o_1)	$[\mathsf{pname}\colon \langle\{\mathsf{Lady\text{-}Fern}, \mathsf{Ostrich\text{-}Fern}\}, \mathrm{u}, \mathrm{u}\rangle,$ $\mathsf{rain}\colon \langle\{25, \ldots, 30\}, \mathrm{u}, \mathrm{u}\rangle,$ $\mathsf{pname2}\colon \langle\{\mathsf{Lady\text{-}Fern}, \mathsf{Ostrich\text{-}Fern}\}, \mathrm{u}, \mathrm{u}\rangle,$ $\mathsf{rain2}\colon \langle\{25, \ldots, 30\}, \mathrm{u}, \mathrm{u}\rangle]$
(o_2, o_1)	$[\mathsf{pname}\colon \langle\{\mathsf{Cuban\text{-}Basil}, \mathsf{Lemon\text{-}Basil}\}, \mathrm{u}, \mathrm{u}\rangle,$ $\mathsf{rain}\colon \langle\{20, \ldots, 30\}, \mathrm{u}, \mathrm{u}\rangle,$ $\mathsf{pname2}\colon \langle\{\mathsf{Lady\text{-}Fern}, \mathsf{Ostrich\text{-}Fern}\}, \mathrm{u}, \mathrm{u}\rangle,$ $\mathsf{rain2}\colon \langle\{25, \ldots, 30\}, \mathrm{u}, \mathrm{u}\rangle]$
(o_3, o_1)	$[\mathsf{pname}\colon \langle\{\mathsf{Mint}\}, \mathrm{u}, \mathrm{u}\rangle,$ $\mathsf{rain}\colon \langle\{20\}, \mathrm{u}, \mathrm{u}\rangle,$ $\mathsf{pname2}\colon \langle\{\mathsf{Lady\text{-}Fern}, \mathsf{Ostrich\text{-}Fern}\}, \mathrm{u}, \mathrm{u}\rangle,$ $\mathsf{rain2}\colon \langle\{25, \ldots, 30\}, \mathrm{u}, \mathrm{u}\rangle]$

Table 9: π and ν resulting from cartesian product (partial view)

Definition 7.20 (join-compatible POB-schemas) Two POB-schemas $\mathrm{S}_1 = (\mathcal{C}_1, \sigma_1, \Rightarrow_1, \mathrm{me}_1, \wp_1)$ and $\mathrm{S}_2 = (\mathcal{C}_2, \sigma_2, \Rightarrow_2, \mathrm{me}_2, \wp_2)$ are *join-compatible* iff $R(\mathrm{S}_1)$ and $R(\mathrm{S}_2)$ are disjoint and $\sigma_1(c_1).A = \sigma_2(c_2).A$ for all classes $c_1 \in \mathcal{C}_1$, $c_2 \in \mathcal{C}_2$ and attributes A defined for both $\sigma_1(c_1)$ and $\sigma_2(c_2)$.

Definition 7.21 (join of POB-schemas) Let $\mathrm{S}_1 = (\mathcal{C}_1, \sigma_1, \Rightarrow_1, \mathrm{me}_1, \wp_1)$ and $\mathrm{S}_2 = (\mathcal{C}_2, \sigma_2, \Rightarrow_2, \mathrm{me}_2, \wp_2)$ be two join-compatible POB-schemas, and let $R_1 = R(\mathrm{S}_1)$ and $R_2 = R(\mathrm{S}_2)$. The *join* of S_1 and S_2, denoted $\mathrm{S}_1 \bowtie \mathrm{S}_2$, is the POB-schema $\mathrm{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$, where $\mathcal{C}$, $\Rightarrow$, me, and $\wp$ are as in the definition of $\mathrm{S} = \mathrm{S}_1 \times \mathrm{S}_2$ (see Definition 7.17), and σ is defined as follows:

- For all $c \in \mathcal{C}$, the tuple type $\sigma(c) = [A_1\colon \tau_1, \ldots, A_l\colon \tau_l]$ contains exactly all $A_i : \tau_i$ that belong to either the tuple type $\sigma_1(c[R_1])$ or the tuple type $\sigma_2(c[R_2])$.

Theorem 7.4 *Let S_1 and S_2 be two join-compatible POB-schemas. If S_1 and S_2 are consistent, then $\mathrm{S}_1 \bowtie \mathrm{S}_2$ is consistent.*

For the join of two probabilistic tuples values ptv_1 and ptv_2, we need to combine the two values of a common attribute A_i to a single value for the result. This is done through conjunction of the probabilistic triples representing these values, along the following definition.

Definition 7.22 (conjunction strategies on probabilistic triples) Let (V', α', β') and (V'', α'', β'') be two probabilistic triples and let $\otimes$ be a probabilistic conjunction strategy.

We define $(V', \alpha', \beta') \otimes (V'', \alpha'', \beta'')$ as the probabilistic triple (V, α, β) with:

- $V = \{v \in V' \cap V'' \mid [\alpha'(v), \beta'(v)] \otimes [\alpha''(v), \beta''(v)] \neq [0, 0]\}$.
- $[\alpha(v), \beta(v)] = [\alpha'(v), \beta'(v)] \otimes [\alpha''(v), \beta''(v)]$ for all $v \in V$.

Note that impossible values v in $V' \cap V''$ (having probability 0) are excluded from V as they are implicitly represented by the CWA. The outcome $pt = pt_1 \otimes pt_2$ is well-defined only if pt is consistent, which requires that $\sum_{v \in V} \beta(v) \geq 1$. When an inconsistency arises, we flag an error.

Definition 7.23 (join of probabilistic tuple values) Let ptv_1 and ptv_2 be two probabilistic tuple values over the sets of attributes $\mathbf{A}_1$ and $\mathbf{A}_2$, respectively, such that for all $A \in \mathbf{A}_1 \cap \mathbf{A}_2$, the values $ptv_1.A$ and $ptv_2.A$ are of the same type. Let $\otimes$ be a probabilistic conjunction strategy. The *join* of ptv_1 and ptv_2 under $\otimes$, denoted $ptv_1 \bowtie_\otimes ptv_2$, is the probabilistic tuple value ptv over $\mathbf{A}_1 \cup \mathbf{A}_2$ defined by:

- $ptv.A = ptv_1.A$ for all attributes $A \in \mathbf{A}_1 - \mathbf{A}_2$.
- $ptv.A = ptv_2.A$ for all attributes $A \in \mathbf{A}_2 - \mathbf{A}_1$.
- $ptv.A = ptv_1.A \otimes ptv_2.A$ for all attributes $A \in \mathbf{A}_1 \cap \mathbf{A}_2$.

Note that, for any probabilistic conjunction strategy $\otimes$, $ptv_1 \bowtie_\otimes ptv_2 = ptv_2 \bowtie_\otimes ptv_1$ holds, i.e., the join of probabilistic tuples values is commutative.

Example 7.19 *Let us consider the following two probabilistic tuple values:*

$$\begin{aligned} ptv_1 &= [A\colon \langle \{a, b\}, 0.6\,\mathrm{u}, 1.4\,\mathrm{u}\rangle,\ B\colon \langle \{a, c\}, 0.7\,\mathrm{u}, 1.3\,\mathrm{u}\rangle] \\ ptv_2 &= [A\colon \langle \{a, b, c\}, 0.3\,\mathrm{u}, 2.4\,\mathrm{u}\rangle,\ C\colon \langle \{c, d\}, 0.4\,\mathrm{u}, 1.6\,\mathrm{u}\rangle] \end{aligned}$$

The join *of ptv_1 and ptv_2 under independence is given as follows:*

$$ptv_1 \bowtie_{\otimes_{in}} ptv_2 = [A\colon \langle \{a, b\}, 0.06\,\mathrm{u}, 1.12\,\mathrm{u}\rangle,\ B\colon \langle \{a, c\}, 0.7\,\mathrm{u}, 1.3\,\mathrm{u}\rangle,\ C\colon \langle \{c, d\}, 0.4\,\mathrm{u}, 1.6\,\mathrm{u}\rangle].$$

Definition 7.24 (join of POB-instances) Let $\mathbf{I}_1 = (\pi_1, \nu_1)$ and $\mathbf{I}_2 = (\pi_2, \nu_2)$ be two POB-instances over the join-compatible POB-schemas $\mathbf{S}_1 = (\mathcal{C}_1, \sigma_1, \Rightarrow_1, \mathrm{me}_1, \wp_1)$ and $\mathbf{S}_2 = (\mathcal{C}_2, \sigma_2, \Rightarrow_2, \mathrm{me}_2, \wp_2)$, respectively, and let $R_1 = R(\mathbf{S}_1)$ and $R_2 = R(\mathbf{S}_2)$. Let $\mathbf{A}_1$ and $\mathbf{A}_2$ be the sets of top-level attributes of $\mathbf{S}_1$ and $\mathbf{S}_2$, respectively. Let $\otimes$ be a probabilistic conjunction strategy. The *join* of $\mathbf{I}_1$ and $\mathbf{I}_2$ under $\otimes$, denoted $\mathbf{I}_1 \bowtie_\otimes \mathbf{I}_2$, is the POB-instance (π, ν) over the POB-schema $\mathbf{S}_1 \bowtie \mathbf{S}_2$, where:

- $\pi(c) = \{(o_1, o_2) \in \pi_1(c[R_1]) \times \pi_2(c[R_2]) \mid$ for all $A \in \mathbf{A}_1 \cap \mathbf{A}_2$:
  if $(\nu_1(o_1) \bowtie_\otimes \nu_2(o_2)).A = \langle V, \alpha, \beta\rangle$, then $V \neq \emptyset\}$, for all $c \in \mathcal{C}_1 \times \mathcal{C}_2$.
- $\nu(o) = \nu_1(o[R_1]) \bowtie_\otimes \nu_2(o[R_2])$, for all $o \in \pi(\mathcal{C})$.

7.6 Intersection, Union, and Difference

In this section, we define the classical set operations of *intersection*, *union*, and *difference* for two POB-instances over the same schema.

The definition of intersection is intuitive: common objects are selected, and their respective attribute values are combined by conjunction. We introduce the following notion.

Definition 7.25 (intersection of probabilistic tuple values) Let ptv_1 and ptv_2 be probabilistic tuple values over the same set of attribute names $\mathbf{A}$ and let $\otimes$ be a probabilistic conjunction strategy. The *intersection* of ptv_1 and ptv_2 under $\otimes$, denoted $ptv_1 \cap_\otimes ptv_2$, is the probabilistic tuple value ptv over $\mathbf{A}$ defined by $ptv.A = ptv_1.A \otimes ptv_2.A$ for all $A \in \mathbf{A}$.

Definition 7.26 (intersection of POB-instances) Let $\mathbf{I}_1 = (\pi_1, \nu_1)$ and $\mathbf{I}_2 = (\pi_2, \nu_2)$ be two POB-instances over the same POB-schema $\mathbf{S}$ and let $\otimes$ be a probabilistic conjunction strategy. The *intersection* of $\mathbf{I}_1$ and $\mathbf{I}_2$ under $\otimes$, denoted $\mathbf{I}_1 \cap_\otimes \mathbf{I}_2$, is the POB-instance (π, ν) over $\mathbf{S}$, where:

- $\pi(c) = \pi_1(c) \cap \pi_2(c)$.
- $\nu(o) = \nu_1(o) \cap_\otimes \nu_2(o)$.

The union of two POB-instances is defined in the same spirit as their intersection.

Definition 7.27 (disjunction strategies on probabilistic triples) Let $pt_1 = (V', \alpha', \beta')$ and $pt_2 = (V'', \alpha'', \beta'')$ be two probabilistic triples and let $\oplus$ be a probabilistic disjunction strategy. Then $pt_1 \oplus pt_2$ is the probabilistic triple $pt = (V, \alpha, \beta)$, where:

- $V = V' \cup V''$.
- $[\alpha(v), \beta(v)] = \begin{cases} [\alpha'(v), \beta'(v)] & \text{if } v \in V' - V'' \\ [\alpha''(v), \beta''(v)] & \text{if } v \in V'' - V' \\ [\alpha'(v), \beta'(v)] \oplus [\alpha''(v), \beta''(v)] & \text{if } v \in V' \cap V''. \end{cases}$

As in the case of conjunction, the outcome pt of $pt_1 \oplus pt_2$ is only defined if pt is consistent, which requires that $\sum_{v \in V} \alpha(v) \leq 1$. A violation of this condition indicates incorrect data or improper application of the disjunction strategy $\oplus$. Again, this is flagged.

Definition 7.28 (union of probabilistic tuple values) Let ptv_1 and ptv_2 be two probabilistic tuple values over the same set of attribute names $\mathbf{A}$ and let $\oplus$ be a probabilistic disjunction strategy. The *union* of ptv_1 and ptv_2 under $\oplus$, denoted $ptv_1 \cup_\oplus ptv_2$, is the probabilistic tuple value ptv over $\mathbf{A}$ defined by $ptv.A = ptv_1.A \oplus ptv_2.A$ for all $A \in \mathbf{A}$.

Definition 7.29 (union of POB-instances) Let $\mathbf{I}_1 = (\pi_1, \nu_1)$ and $\mathbf{I}_2 = (\pi_2, \nu_2)$ be two POB-instances over the same POB-schema $\mathbf{S}$ such that $\pi_1(c_1) \cap \pi_2(c_2) = \emptyset$ for all pairs of distinct classes $c_1, c_2 \in \mathcal{C}$. Let $\oplus$ be a probabilistic disjunction strategy. The *union* of $\mathbf{I}_1$ and $\mathbf{I}_2$ under $\oplus$, denoted $\mathbf{I}_1 \cup_\oplus \mathbf{I}_2$, is defined as the POB-instance (π, ν) over $\mathbf{S}$, where:

- $\pi(c) = \pi_1(c) \cup \pi_2(c)$.
- $\nu(o) = \begin{cases} \nu_1(o) & \text{if } o \in \pi_1(\mathcal{C}) - \pi_2(\mathcal{C}) \\ \nu_2(o) & \text{if } o \in \pi_2(\mathcal{C}) - \pi_1(\mathcal{C}) \\ \nu_1(o) \cup_\oplus \nu_2(o) & \text{if } o \in \pi_1(\mathcal{C}) \cap \pi_2(\mathcal{C}). \end{cases}$

Finally, we consider the *difference* of two POB-instances. For this, we use the notion of a difference strategy for probabilistic tuple values.

Definition 7.30 (difference strategies on probabilistic triples) Let $pt_1 = (V', \alpha', \beta')$ and $pt_2 = (V'', \alpha'', \beta'')$ be two probabilistic triples and let $\ominus$ be a probabilistic difference strategy.

We define $pt_1 \ominus pt_2$ as the probabilistic triple $pt = (V, \alpha, \beta)$ with:

- $V = V' - \{v \in V' \cap V'' \mid [\alpha'(v), \beta'(v)] \ominus [\alpha''(v), \beta''(v)] = [0, 0\}]$.
- $[\alpha(v), \beta(v)] = \begin{cases} [\alpha'(v), \beta'(v)] & \text{if } v \in V - V'' \\ [\alpha'(v), \beta'(v)] \ominus [\alpha''(v), \beta''(v)] & \text{if } v \in V \cap V''. \end{cases}$

Definition 7.31 (difference of probabilistic tuple values) Let ptv_1 and ptv_2 be two probabilistic tuple values over the same set of attribute names $\mathbf{A}$, and let $\ominus$ be a probabilistic difference strategy. The *difference* of ptv_1 and ptv_2 under $\ominus$, denoted $ptv_1 -_\ominus ptv_2$ is the probabilistic tuple value ptv over $\mathbf{A}$ defined by $ptv.A = ptv_1.A \ominus ptv_2.A$ for all $A \in \mathbf{A}$.

Definition 7.32 (difference of POB-instances) Let $\mathbf{I}_1 = (\pi_1, \nu_1)$ and $\mathbf{I}_2 = (\pi_2, \nu_2)$ be POB-instances over the same POB-schema $\mathbf{S}$ and let $\ominus$ be a probabilistic difference strategy. The *difference* of $\mathbf{I}_1$ and $\mathbf{I}_2$ under $\ominus$, denoted $\mathbf{I}_1 -_\ominus \mathbf{I}_2$, is defined as the POB-instance (π, ν) over $\mathbf{S}$, where:

- $\pi(c) = \pi_1(c)$.
- $\nu(o) = \begin{cases} \nu_1(o) & \text{if } o \in \pi_1(\mathcal{C}) - \pi_2(\mathcal{C}) \\ \nu_1(o) -_\ominus \nu_2(o) & \text{if } o \in \pi_1(\mathcal{C}) \cap \pi_2(\mathcal{C}). \end{cases}$

8 POB Algebra: Equivalence Results

In this section, we derive some results on equivalences which hold in our POB-algebra. We focus here on equivalences similar to well-known equivalences in the context of classical relational algebra. The list of equivalences is by no means complete, but shows that query optimization is possible along similar lines in classical relational algebra [1]. Our first result says that the selections may be reordered.

Theorem 8.1 *Let $\mathbf{I} = (\pi, \nu)$ be a POB-instance over the POB-schema $\mathbf{S}$. Let ϕ_1 and ϕ_2 be two probabilistic selection conditions. Then*

$$\sigma_{\phi_1}(\sigma_{\phi_2}(\mathbf{I})) \;=\; \sigma_{\phi_2}(\sigma_{\phi_1}(\mathbf{I})) \;=\; \sigma_{\phi_1 \wedge \phi_2}(\mathbf{I}), \tag{1}$$

where the last expression assumes that ϕ_1 and ϕ_2 have the same object variable.

Our next result says two things: first that the projections may be reordered and second, that projections may be pushed through selections under appropriate contributions.

Theorem 8.2 *Let* $\mathbf{I}$ *be a POB-instance over the POB-schema* $\mathbf{S}$*. Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be sets of attributes, and let* ϕ *be a probabilistic selection condition in which all path expressions start with attribute names from* $\mathbf{A}$*. Then,*

$$\Pi_{\mathbf{A}}(\Pi_{\mathbf{B}}(\mathbf{I})) = \Pi_{\mathbf{B}}(\Pi_{\mathbf{A}}(\mathbf{I})) \tag{2}$$

$$\Pi_{\mathbf{A}}(\sigma_{\phi}(\mathbf{I})) = \sigma_{\phi}(\Pi_{\mathbf{A}}(\mathbf{I}))\,. \tag{3}$$

The next result, which states that selections and projections can be pushed through the renaming operator, requires some notation. For any renaming condition $N : \vec{\mathbf{B}} \leftarrow \vec{\mathbf{C}}$, the *inverse* of N, denoted by N^{-1}, is the renaming condition $\vec{\mathbf{C}} \leftarrow \vec{\mathbf{B}}$. Furthermore, the notation $\delta_N(X)$ stands for the result of applying the renaming N on the formal object X.

Theorem 8.3 *Let* $\mathbf{I}$ *be a POB-instance over the POB-schema* $\mathbf{S}$*, and let* N *be a renaming condition for* $\mathbf{S}$*. Let* ϕ *be a probabilistic selection condition and let* $\mathbf{A}$ *be a set of attributes. Then*

$$\sigma_{\phi}(\delta_N(\mathbf{I})) = \delta_N(\sigma_{\delta_{N^{-1}}(\phi)}(\mathbf{I})) \tag{4}$$

$$\Pi_{\mathbf{A}}(\delta_N(\mathbf{I})) = \delta_N(\Pi_{\delta_{N^{-1}}(\mathbf{A})}(\mathbf{I}))\,. \tag{5}$$

The following theorem shows that joins are always associative and commutative, regardless of what conjunction strategy is used in the join. In addition, selects may be pushed "through" a join by appropriately splitting the selection condition and the same is true of projections.

Theorem 8.4 *Let* $\mathbf{S}_1$*,* $\mathbf{S}_2$*, and* $\mathbf{S}_3$ *be pairwise join-compatible POB-schemas and let* $\mathbf{I}_1$*,* $\mathbf{I}_2$*, and* $\mathbf{I}_3$ *be POB-instances over* $\mathbf{S}_1$*,* $\mathbf{S}_2$*, and* $\mathbf{S}_3$*, respectively. Let* $\otimes$ *be a probabilistic conjunction strategy. Let* ϕ_1*,* ϕ_2*, and* ϕ_3 *be probabilistic selection conditions such that* ϕ_1 *and* ϕ_2 *involve only attributes from* $\mathbf{A}_1 - \mathbf{A}_2$ *and* $\mathbf{A}_2 - \mathbf{A}_1$*, respectively, where* $\mathbf{A}_1$ *and* $\mathbf{A}_2$ *denote the sets of top-level attributes of* $\mathbf{S}_1$ *and* $\mathbf{S}_2$*, respectively. Let* $\mathbf{B}$ *be a set of attributes and define* $\mathbf{B}_1 = (\mathbf{B} \cup \mathbf{A}_2) \cap \mathbf{A}_1$ *and* $\mathbf{B}_2 = (\mathbf{B} \cup \mathbf{A}_1) \cap \mathbf{A}_2$*. Then*

$$\mathbf{I}_1 \bowtie_{\otimes} \mathbf{I}_2 = \mathbf{I}_2 \bowtie_{\otimes} \mathbf{I}_1 \tag{6}$$

$$(\mathbf{I}_1 \bowtie_{\otimes} \mathbf{I}_2) \bowtie_{\otimes} \mathbf{I}_3 = \mathbf{I}_1 \bowtie_{\otimes} (\mathbf{I}_2 \bowtie_{\otimes} \mathbf{I}_3) \tag{7}$$

$$\sigma_{\phi_1 \wedge \phi_2 \wedge \phi_3}(\mathbf{I}_1 \bowtie_{\otimes} \mathbf{I}_2) = \sigma_{\phi_3}(\sigma_{\phi_1}(\mathbf{I}_1) \bowtie_{\otimes} \sigma_{\phi_2}(\mathbf{I}_2)) \tag{8}$$

$$\Pi_{\mathbf{B}}(\mathbf{I}_1 \bowtie_{\otimes} \mathbf{I}_2) = \Pi_{\mathbf{B}}(\Pi_{\mathbf{B}_1}(\mathbf{I}_1) \bowtie_{\otimes} \Pi_{\mathbf{B}_2}(\mathbf{I}_2))\,. \tag{9}$$

Note that in classical relational databases, Equivalence (8) remains true if ϕ_1 and ϕ_2 access common attributes of $\mathbf{A}_1$ and $\mathbf{A}_2$. This is no longer guaranteed for POBs, as the join may change the value of common attributes. As cartesian product is a special case of join, we obtain the following corollary to Theorem 8.4.

Corollary 8.5 *Let* S_1, S_2, *and* S_3 *be pairwise cartesian-product-compatible POB-schemas and let* $\mathbf{I}_1$, $\mathbf{I}_2$, *and* $\mathbf{I}_3$ *be POB-instances over* S_1, S_2, *and* S_3, *respectively. Let* ϕ_1, ϕ_2, *and* ϕ_3 *be probabilistic selection conditions such that* ϕ_1 *and* ϕ_2 *involve only attributes from the sets of top-level attributes* $\mathbf{A}_1$ *and* $\mathbf{A}_2$ *of* S_1 *and* S_2, *respectively. Let* $\mathbf{B}$ *be a set of attributes and let* $\mathbf{B}_1 = \mathbf{B} \cap \mathbf{A}_1$ *and* $\mathbf{B}_2 = \mathbf{B} \cap \mathbf{A}_2$. *Then*

$$\mathbf{I}_1 \times \mathbf{I}_2 = \mathbf{I}_2 \times \mathbf{I}_1 \tag{10}$$

$$(\mathbf{I}_1 \times \mathbf{I}_2) \times \mathbf{I}_3 = \mathbf{I}_1 \times (\mathbf{I}_2 \times \mathbf{I}_3) \tag{11}$$

$$\sigma_{\phi_1 \wedge \phi_2 \wedge \phi_3}(\mathbf{I}_1 \times \mathbf{I}_2) = \sigma_{\phi_3}(\sigma_{\phi_1}(\mathbf{I}_1) \times \sigma_{\phi_2}(\mathbf{I}_2)) \tag{12}$$

$$\Pi_{\mathbf{B}}(\mathbf{I}_1 \times \mathbf{I}_2) = \Pi_{\mathbf{B}_1}(\mathbf{I}_1) \times \Pi_{\mathbf{B}_2}(\mathbf{I}_2)\,. \tag{13}$$

Theorem 8.6 *Let* $\mathbf{I}_1$, $\mathbf{I}_2$, *and* $\mathbf{I}_3$ *be POB-instances over the same POB-schema* S. *Let* $\otimes / \oplus / \ominus$ *be a probabilistic conjunction/disjunction/difference strategy and let* $\mathbf{A}$ *be a set of attributes. Then,*

$$\mathbf{I}_1 \cap_\otimes \mathbf{I}_2 = \mathbf{I}_2 \cap_\otimes \mathbf{I}_1 \tag{14}$$

$$(\mathbf{I}_1 \cap_\otimes \mathbf{I}_2) \cap_\otimes \mathbf{I}_3 = \mathbf{I}_1 \cap_\otimes (\mathbf{I}_2 \cap_\otimes \mathbf{I}_3) \tag{15}$$

$$\mathbf{I}_1 \cup_\oplus \mathbf{I}_2 = \mathbf{I}_2 \cup_\oplus \mathbf{I}_1 \tag{16}$$

$$(\mathbf{I}_1 \cup_\oplus \mathbf{I}_2) \cup_\oplus \mathbf{I}_3 = \mathbf{I}_1 \cup_\oplus (\mathbf{I}_2 \cup_\oplus \mathbf{I}_3) \tag{17}$$

$$\Pi_{\mathbf{A}}(\mathbf{I}_1 \cap_\otimes \mathbf{I}_2) = \Pi_{\mathbf{A}}(\mathbf{I}_1) \cap_\otimes \Pi_{\mathbf{A}}(\mathbf{I}_2) \tag{18}$$

$$\Pi_{\mathbf{A}}(\mathbf{I}_1 \cup_\oplus \mathbf{I}_2) = \Pi_{\mathbf{A}}(\mathbf{I}_1) \cup_\oplus \Pi_{\mathbf{A}}(\mathbf{I}_2) \tag{19}$$

$$\Pi_{\mathbf{A}}(\mathbf{I}_1 -_\ominus \mathbf{I}_2) = \Pi_{\mathbf{A}}(\mathbf{I}_1) -_\ominus \Pi_{\mathbf{A}}(\mathbf{I}_2)\,. \tag{20}$$

Note that literally taken, Equations (20) and (18) are not true for relational databases. The reason is that we use oids for objects in POBs, while relational databases contain simply values.

9 Implementation

We have implemented a prototype of a distributed probabilistic object database system. The server (POB server) runs on top of ObjectStore (Version 6.0), and is implemented in SUN-C++ (Version 4.2). A thin client for handling database transactions is implemented using GNU-C++ (Version 2.8.1).

9.1 POB Server

The POB-server is a collection manager of POB-schemas, where each POB-schema consists of a set of POB-classes and their associated POB-object instances. There are two types of POB-schemas that the server manages: the persistent schemas, which correspond to the permanent data that the database keeps, and temporary schemas, which maintain intermediate schemas created during transaction processing.

The **probability interpreter** contains a collection of functions for computing probabilistic conjunction and disjunction strategies. It holds, in addition, a library of distribution functions for manipulating probabilistic tuple values associated with objects in the database.

The **POB-schema** class maintains an inheritance probability table (the probability assignment $\wp$ in Definition 5.4). The class contains methods to add and remove POB classes, add and remove POB-objects, and POB-classes, and to retrieve POB-objects and POB-classes. In addition, given two classes c_1, c_2, there is a method that computes the probability that c_1 is a subclass of c_2.

Each object in the **POB** class has a name, a collection of attributes and their associated types, and a collection of parent POB-class names along with their associated probability assignments. Methods associated with POB-class objects provide abilities to establish attribute/type information, parent POB-class/probability assignments, adding and removing POB-objects from the POB-class, and various self-replicating functions that are useful for query processing.

POB objects contain an object name, the oid, a collection of probabilistic tuple values, and a POB-class pointer which points to the POB-class of which it is an instance. The POB-class pointer is provided for fast access to class-level information: attributes, types, parents, etc. Methods associated with POB-objects include functions for setting probabilistic tuple values and various self-replicating functions to facilitate query processing.

The **POB Server** handles client requests. It contains a pointer to an ObjectStore database which maintains persistence for the POB-server. The POB Server includes methods for: connecting to a database, disconnecting from a database, creating and removing schemas, creating and removing classes, creating and removing objects, computing the probability that an object is a member of class c_1 given that it is a member of class c_2, computing the probabilistic extent of a class, checking if an object satisfies a given probabilistic selection condition, executing an arbitrary query in the probabilistic object algebra, and a variety of printing functions.

Note that each method may not correspond to a logical unit of work — in this case a request. In some instances, several requests are handled within one method while in other instances, a single request is handled through a combination of methods.

9.2 Experiments

Using the POB Server, we have conducted a set of experiments to assess the various equivalences described in Section 8 as well as to assess the performance of selection. We do not describe all the experiments we conducted (due to space reasons), but only a few sample experiments are listed below. The limiting factor in all experiments was the size of the "largest intermediate schema." This is the number of objects in the largest schema encountered when executing the query. For a selection query, this is just the number of objects in the POB-instance on which the selection is performed. In the case of a join/cartesian product, this is the product of the sizes of the POB-instances being joined (or whose Cartesian product is being computed). In the experiments involving Equations 8, 12 and 13 described below, we varied the number of objects in the largest intermediate schema from 0 to 270,000 objects and measured the times taken (on a Sun Ultra 10 workstation) for both the left side and the right side of the rewrite rules in question. The idea was to see whether the left side of a rule should be rewritten to the right side or the other way round.

Effectiveness of Equation 8. Our first experiment evaluated the effectiveness of pushing selections into joins (Theorem 8.4). Figure 9.2(a) shows what happens if the selectivity is varied using independence. It is easy to see that the right side of this equation pays off in a huge way and that as the selectivity decreases (i.e., fewer and fewer objects are selected), more and more objects can be efficiently handed. For instance, with 20% selectivity, 270,000 objects in the largest intermediate schema can be computed in about 30 seconds.

Figure 9.2(b) shows the effect of evaluating the right side of Equation 8 with different probabilistic strategies. We see that precisely which strategy is used has very little impact on the computation time, disputing the oft-held folk claim that assuming independence of events is necessary for efficiency reasons.

Effectiveness of Equation 12. We conducted experiments similar to those described above with Equation 12. Figure 9.2(c) shows the result of testing — it shows that pushing selections into a cartesian product may save up to 80–90% of the time and this saving increases as the number of objects increases.

Effectiveness of Equation 13. We conducted experiments similar to those described above with Equation 13. Figure 9.2(d) shows the result of testing — it shows that pushing projections into a cartesian product does not help very much. The reason for this is because projection does not reduce the number of objects.

Effectiveness of Selection. We also conducted some experiments on the effectiveness of selection on POB-schemas of sizes between 3000 and 10000 objects. In the experiment, we executed queries of the form "Select x from schema e where $x.D > val$". Figure 9.2(e) shows the result when two different selectivities are used — 50% (i.e. half the objects satisfy the selection condition) and 30% (i.e. 30% of the objects satisfy the selection condition). We also tested what happens when we consider *membership* selection queries of the form "Select x from schema e where (x is a member of class C)". Figure 9.2(f) shows the result of this query with two different selectivities. Note that the queries generally exhibit linear behavior w.r.t. the number of objects. In addition, membership queries are computationally more expensive than simple inequality queries.

10 Related Work

Our work has been inspired by the prior work of Kornatzky and Shimony [23] who describe a probabilistic object-oriented data model in which, like in our approach, uncertainty in the values of attributes and in the class graph may be represented by probabilities. The main differences between [23] and our approach can be briefly summarized as follows:

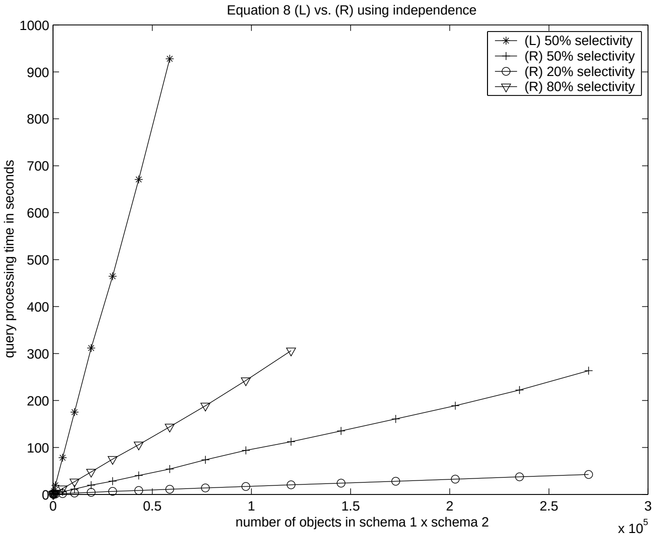
- The work in [23] introduces an object calculus for extracting objects from probabilistic object-oriented databases. This calculus can thus be compared to our selection operation. It is more restrictive in the sense that it only handles probabilities on atomic formulas (which always evaluate to either true or false), while our selection operation also handles probabilities on conjunctions and disjunctions of atomic formulas, using probabilistic conjunction and disjunction strategies. Specifically, we make no independence assumption (which [23] does). On the other hand, their object calculus has quantifiers, which our selection operation does not include. However, it could be easily extended in this direction.

- We also discuss in detail, the algebraic operations of projection, renaming, cartesian product, join, selection, union, intersection, and difference. As they were developing a calculus, [23] does not deal with this.

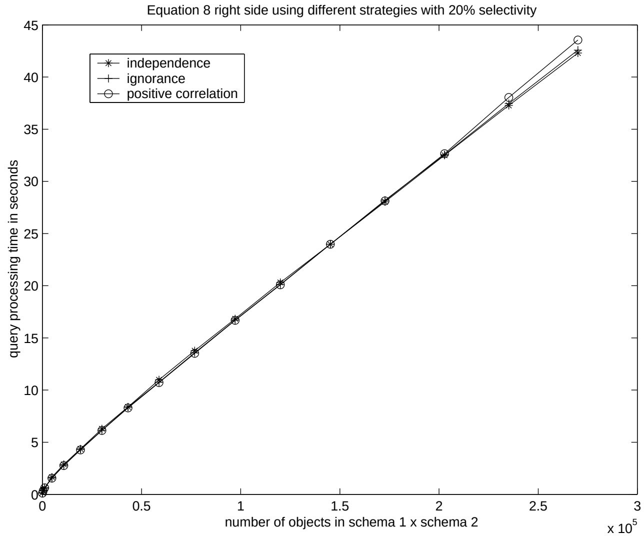
- We introduce for the first time, results on query equivalences in probabilistic object bases, and to our knowledge, our system is the first implementation of a probabilistic object base.

- In [23], the class graph is a directed tree *without multiple inheritance*. Moreover, incomparable classes are always disjoint. In contrast, in our approach, the class graph may be any directed acyclic graph, thus allowing multiple inheritance. Furthermore, the disjointness of classes can be expressed in a flexible way by grouping them into partition clusters. The consistency of schema declarations is guaranteed for a large subclass extending directed trees.

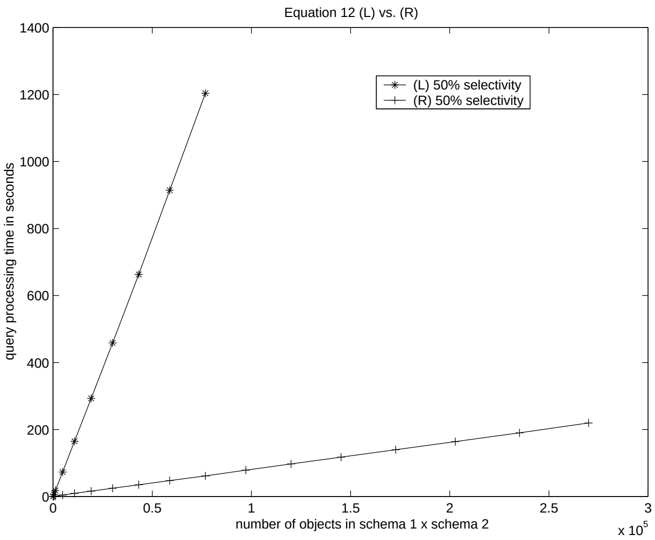
- Kornatzky and Shimony assume a precise probability distribution on the set of all possible values of an attribute (including a null value $\perp$ that represents the inapplicability of an attribute). Our approach, in contrast, just requires an interval range for probability distributions. Furthermore, objects occurring as attribute values are given special treatment in [23]; our model can be extended in this respect.


Equation 8 (L) vs. (R) using independence
(L) 50% selectivity
(R) 50% selectivity
(R) 20% selectivity
(R) 80% selectivity
query processing time in seconds
number of objects in schema 1 x schema 2
x 10^5

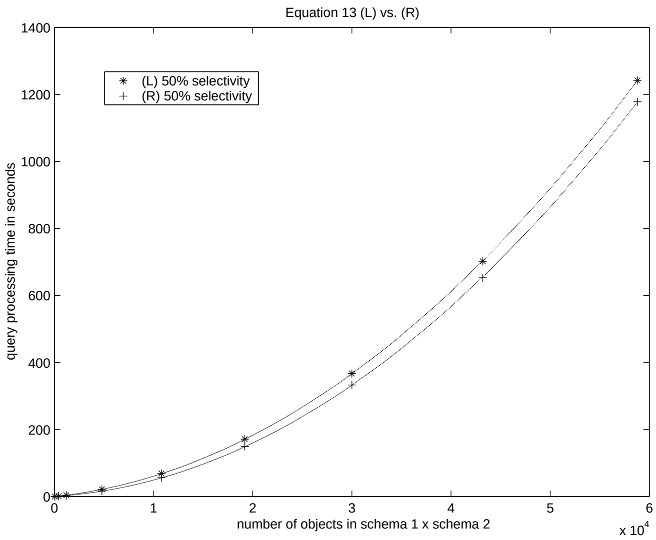
(a)


Equation 8 right side using different strategies with 20% selectivity
independence
ignorance
positive correlation
query processing time in seconds
number of objects in schema 1 x schema 2
x 10^5

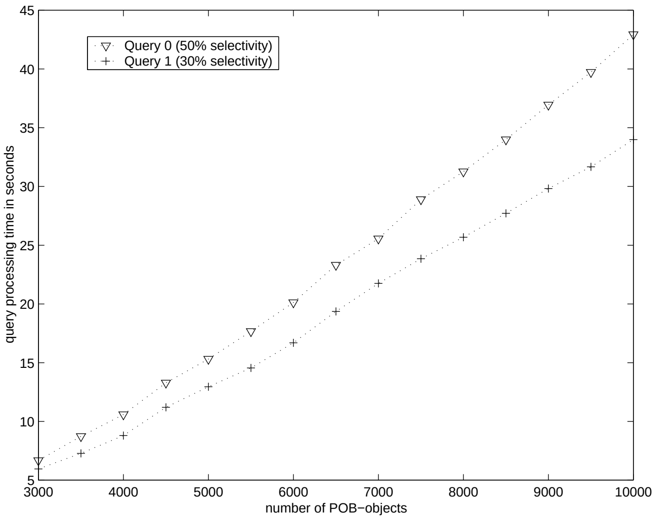
(b)


Equation 12 (L) vs. (R)
(L) 50% selectivity
(R) 50% selectivity
query processing time in seconds
number of objects in schema 1 x schema 2
x 10^5

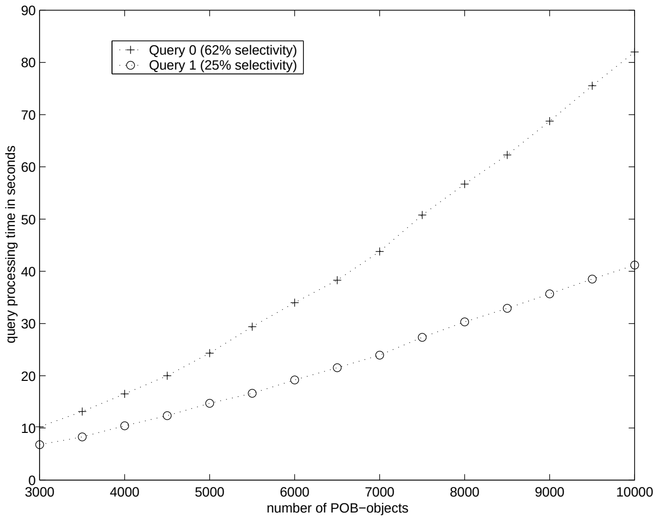
(c)


Equation 13 (L) vs. (R)
(L) 50% selectivity
(R) 50% selectivity
query processing time in seconds
number of objects in schema 1 x schema 2
x 10^4

(d)


Query 0 (50% selectivity)
Query 1 (30% selectivity)
query processing time in seconds
number of POB−objects

(e)


Query 0 (62% selectivity)
Query 1 (25% selectivity)
query processing time in seconds
number of POB−objects

(f)

- In [23], the probabilistic extent of a class is derived from statistical and subjective probabilities. Since, in general, inconsistency may arise, the notion of *cutsets* of classes is introduced there. The probabilistic extent of a class is then given by statistical probabilities in the class hierarchy and by subjective probabilities with respect to a cutset. Our probabilistic extent, in contrast, is just derived from statistical probabilities and classical class membership. We thus avoid all the problems that come along with mixing up statistical and subjective probabilities.

A step towards the model proposed in the present paper is an extension of the relational model allowing complex values [12] with probabilities. However, the model in [12] has no a class hierarchy and, in particular, inheritance is not addressed. Thus, it has no features of an object oriented system, and is essentially in the group of probabilistic relational database models, which we discuss next.

ProbView [24] is a probabilistic relation database model which generalizes various approaches (like, for example, [3, 6]). Cavallo and Pittarelli's important paper [6] views relations in a (flat) relational database as probability distribution functions, where tuples in the same relation are viewed as pairwise disjoint events whose probabilities sum up to 1. Drawbacks of this approach have been pointed out in [8]. An extension of the model using probability intervals, which are viewed as constraints on the probabilities, is reviewed in [28]. Barbará et al. [3] consider a probabilistic extension to the relational model, in which imprecise attributes are modeled as probability distributions over finite sets of values. No probabilities can be assigned to outmost tuples. Their approach assumes that key attributes are deterministic (have probability 1) and that non-key attributes in different relations are independent. As pointed out in [3], "lossy" joins are possible in this model.

Another important probabilistic database model is that of Dey and Sarkar [8], which assigns each tuple in a (flat) relational database a probability value in a special attribute. Based on [8], a probabilistic extension to SQL is developed in [9]. The classical relational operations are in [8] defined adopting different assumptions on the relationship between tuples; in particular, join assumes independence; union and difference assume positive correlation; and compaction assumes disjointness or positive correlation. Our model is far more general.

Fuhr and Rölleke [14] consider a probabilistic version of NF2 relations, extending their approach for flat tuples [15], and define a relational algebra for this model. Probabilities are assigned to tuples and to values of nested tuples (that is, set-valued attributes), which are viewed as events that have an associated event expression. The algebraic operators manipulate tuples by combining value and event expressions appropriately. An intensional semantics is developed in [14] in which probabilities are defined through possible worlds. The evaluation method assumes that in nondeterministic relations (that is, relations with uncertain tuples), joint occurrence of two different values is either always independent or impossible—this is certainly restrictive.

Dyreson and Snodgrass [10] provide a version of SQL to handle temporal indeterminacy, where there is uncertainty about when an event occurs. They use a relational framework and focus on the important case where the space of values over which uncertainty exists is huge.

Kießling and his group [20] developed a framework called DUCK for reasoning with uncertainty. They provide an elegant, logical, axiomatic theory for uncertain reasoning in the presence of rules. In contrast, in our framework, rules are not present; rather, our interest is in extending object database models to handle uncertainty in an algebraic setting.

In an important paper, Lakshmanan and Sadri [26] show how selected probabilistic strategies can be used to extend the previous probabilistic models. Lakshmanan and Shiri [27] show how deductive databases may be parameterized through the use of conjunction and disjunction strategies, an approach also followed by Dekhtyar and Subrahmanian [7]. We have built in this paper upon the important concept of probabilistic conjunction and disjunction strategies, but in an object oriented instead of a logic programming setting.

11 Conclusion

In this paper, we proposed an extension of the relational algebra to handle probabilistic modes of uncertainty in object oriented database systems. More precisely, the main contributions of this paper can be briefly summarized as follows:

1. We presented a formal definition of a probabilistic object base, which extends previous definitions given by Kornatzky and Shimony [23].

2. We gave a formal model theoretic basis for discussing the consistency of POBs, and showed that consistency checking is NP-complete in general. We then defined classes of POBs for which consistency can be checked in polynomial time, and provided efficient algorithms for this task.

3. We developed an algebra that extends the relational algebra to probabilistic object bases. Specifically, this algebra recognizes that probabilities of complex events depend on existing knowledge about dependencies between events, and hence, it allows users to express algebraic queries under appropriate conjunction, disjunction, and difference strategies (which encode such dependence information).

4. We presented a number of equivalence results that may form a set of rewrite rules to be used in query optimization.

5. Our POB framework has been implemented on top of ObjectStore and the VisiBroker ORB.

6. Finally, we conducted a set of experiments on the efficacy of our equivalence results for query rewriting (and hence for query optimization).

Several tasks remain for further work. One is the enhancement of the current prototype by a sophisticated POB-algebra query manager, which optimizes queries by using cost models and rewrite rules as shown in Figure 2. For the front end of the system, it would be well worth developing a probabilistic version of SQL (similar to, for example, Dey and Sakar's language PSQL [9]).

Acknowledgments This work were supported by a TASC/DARPA grant J09301S98061, by the Army Research Laboratory under contract number DAAL01-97-K0135, by an NSF Young Investigator award IRI-93-57756 and an NSF grant CCR9731893, and by the Austrian Science Fund under Project N Z29-INF.

References

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, Reading, 1995.

[2] F. Bancilhon, C. Delobel, and P. Kanellakis, editors. *Building an Object-Oriented Database System: The Story of O_2*. Morgan Kaufmann, Los Altos (CA), 1991.

[3] D. Barbara, H. Garcia-Molina, and D. Porter. The management of probabilistic data. *IEEE Transactions on Knowledge and Data Engineering*, 4(5):387–502, 1992.

[4] E. Bertino and L. Martino. *Object Oriented Database Systems: Concepts and Architectures*. Addison-Wesley, Wokingham, 1993.

[5] G. Boole. *The Laws of Thought*. Macmillan, London, 1854.

[6] R. Cavallo and M. Pittarelli. The theory of probabilistic databases. In *Proceedings of the 13th International Conference on Very Large Databases (VLDB '87)*, pages 71–81. Morgan Kaufmann, 1987.

[7] A. Dekhtyar and V. S. Subrahmanian. Hybrid probabilistic programs. In L. Naish, editor, *Proc. of the 14th International Conference on Logic Programming (ICLP '97)*, pages 391–405. MIT Press, 1997.

[8] D. Dey and S. Sarkar. A probabilistic relational model and algebra. *ACM Transactions on Database Systems*, 21(3):339–369, 1996.

[9] D. Dey and S. Sarkar. PSQL: A query language for probabilistic relational data. *Data & Knowledge Engineering*, 28:107–120, 1998.

[10] C. Dyreson and R. Snodgrass. Supporting valid-time indeterminacy. *ACM Transactions on Database Systems*, 23(1):1–57, 1998.

[11] T. Eiter, J. J. Lu, T. Lukasiewicz, and V. S. Subrahmanian. Probabilistic object bases. Technical Report ?????, Computer Science Department, University of Maryland, November 1999. Available from `http://www.cs.umd.edu/?????`.

[12] T. Eiter, T. Lukasiewicz, and M. Walter. Extension of the relational algebra to probabilistic complex values. In *Proceedings of the International Symposium on Foundations of Information and Knowledge Systems (FoIKS 2000),* Burg (Spreewald), Germany, *LNCS*. Springer, 2000. To appear.

[13] R. Fagin, J. Y. Halpern, and N. Megiddo. A logic for reasoning about probabilities. *Information and Computation*, 87:78–128, 1990.

[14] N. Fuhr and T. Rölleke. A probabilistic NF2 relational algebra for integrated information retrieval and database systems. In *Proceedings of the 2nd World Conference on Integrated Design and Process Technology*, pages 17–30. Society for Design and Process Science, 1996.

[15] N. Fuhr and T. Rölleke. A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM Transactions on Information Systems*, 15(1):32–66, 1997.

[16] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman & Co., 1979.

[17] W. I. Grosky, R. Jain, and R. Mehrotra, editors. *The Handbook of Multimedia Information Management*. Prentice Hall, 1997.

[18] U. Güntzer, W. Kießling, and H. Thöne. New directions for uncertainty reasoning in deductive databases. In *Proceedings of ACM SIGMOD '91*, pages 178–187. ACM Press, 1991.

[19] J. Y. Halpern. An analysis of first-order logics of probability. *Artificial Intelligence*, 46(3):311–350, 1990.

[20] W. Kießling, H. Thöne, and U. Güntzer. Database support for problematic knowledge. In A. Pirotte, C. Delobel, and G. Gottlob, editors, *Proceedings of the 3rd International Conference on Extending Database Technology (EDBT-92)*, volume 580 of *LNCS*, pages 421–436. Springer, 1992.

[21] M. Kifer and A. Li. On the semantics of rule-based expert systems with uncertainty. In M. Gyssens, J. Paredaens, and D. Van Gucht, editors, *Proceedings of the 2nd International Conference on Database Theory (ICDT '88)*, volume 326 of *LNCS*, pages 102–117. Springer, 1988.

[22] W. Kim. *Introduction to Object-Oriented Databases*. MIT Press, Cambridge (MA), 1990.

[23] Y. Kornatzky and S. E. Shimony. A probabilistic object-oriented data model. *Data & Knowledge Engineering*, 12:143–166, 1994.

[24] L. V. S. Lakshmanan, N. Leone, R. Ross, and V. S. Subrahmanian. ProbView: A flexible probabilistic database system. *ACM Transactions on Database Systems*, 22:419–469, 1997.

[25] L. V. S. Lakshmanan and F. Sadri. Modeling uncertainty in deductive databases. In *Proceedings of the 5th International Conference on Database and Expert Systems Applications (DEXA '94)*, volume 856 of *LNCS*, pages 724–733. Springer, 1994.

[26] L. V. S. Lakshmanan and F. Sadri. Probabilistic deductive databases. In *Proceedings of the 1994 International Logic Programming Symposium (ILPS '94)*, pages 254–268. MIT Press, 1994.

[27] L. V. S. Lakshmanan and N. Shiri. A parametric approach to deductive databases with uncertainty. In *Proceedings of the International Workshop on Logic in Databases (LID '96)*, volume 1154 of *LNCS*, pages 61–81. Springer, 1996. (To appear in *IEEE Transactions on Knowledge and Data Engineering*).

[28] M. Pittarelli. An algebra for probabilistic databases. *IEEE Transactions on Knowledge and Data Engineering*, 6(2):293–303, 1994.

Appendix

This appendix contains the proofs of all technical results in the main body of the paper.

Proof of Theorem 5.1. Let S $= (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ be a consistent POB-schema. We first show P1. Suppose that there are two different paths from a class c_1 to another class c_2. Let p and q be the products of the edge probabilities on these two paths. By C1 and C4, every model ε of S satisfies $\varepsilon(c_2) \neq \emptyset$ and $p \cdot |\varepsilon(c_2)| = |\varepsilon(c_1)| = q \cdot |\varepsilon(c_2)|$. Thus, we get $p = q$.

We next prove P2 by contradiction. Let us assume that there exists some cluster $\mathcal{P} \in \bigcup \mathrm{me}(\mathcal{C})$ and that there are two distinct classes $c_1, c_2 \in \mathcal{P}$ that have a common subclass $c \in \mathcal{C}$. Hence, by C2 and C3, every model ε of S satisfies $\varepsilon(c) = \emptyset$. But this contradicts C1. □

Proof of Theorem 5.2. It remains to show NP-hardness. We give a polynomial reduction from the NP-complete problem of graph 3-colorability [16]. Let (V, E) be an undirected graph. We now have to construct a POB-schema S such that (V, E) is 3-colorable iff S is consistent. The main idea is to represent each node $v \in V$ by three pairwise disjoint and nonempty classes d_v^1, d_v^2, and d_v^3, which encode the three colors and the properties of a mapping from V into the color set. Each edge $\{u, v\} \in E$ is then represented by making d_u^1, d_u^2, and d_u^3 disjoint from d_v^1, d_v^2, and d_v^3, respectively.

The technical formalization is quite tricky. The POB-schema S $= (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ is constructed as follows. We initialize $\mathcal{C}$, $\Rightarrow$, and $\wp$ with $\{b, c_1, c_2, d\}$, $\{(c_1, b), (c_2, b), (d, c_1)\}$, and $\{(c_1, b, 1/4), (c_2, b, 3/4), (d, c_1, 1)\}$, respectively, and extend them as follows. We add to $\mathcal{C}$ for each $v \in V$ and $i \in \{0, 1, 2\}$ classes c_v^i and d_v^i, and for each $\{u, v\} \in E$ and $i \in \{0, 1, 2\}$ classes $d_{u,v}^i$ and $d_{v,u}^i$. For each $v \in V$ and $i \in \{0, 1, 2\}$, we add the edges $c_v^i \Rightarrow b$, $d \Rightarrow c_v^i$, $d_v^i \Rightarrow c_v^i$, and $d_v^i \Rightarrow c_2$, and assign them the probabilities $1/2, 1/2, 1/2$, and $1/3$, respectively. Furthermore, for each $d_{u,v}^i \in \mathcal{C}$ where $i \in \{0, 1, 2\}$, we add the edges $d_{u,v}^i \Rightarrow c_u^i$ and $d_{u,v}^i \Rightarrow c_2$, and assign them probabilities $1/2$ and $1/3$, respectively. Finally, let $\sigma(c) = [\,]$ for all $c \in \mathcal{C}$ and me as follows:

$$
\begin{aligned}
\mathrm{me}(b) &= \{\{c_1, c_2\}\} \cup \{\{c_v^i\} \mid v \in V,\ i \in \{0,1,2\}\}, \\
\mathrm{me}(c_1) &= \{\{d\}\}, \\
\mathrm{me}(c_2) &= \{\{d_v^1, d_v^2, d_v^3\} \mid v \in V\} \cup \{\{d_{u,v}^i, d_{v,u}^i\} \mid \{u,v\} \in E,\ i \in \{0,1,2\}\}, \\
\mathrm{me}(c_v^i) &= \{\{c\} \mid c \in \mathcal{C},\ c \Rightarrow c_v^i\} \quad \text{for all } v \in V \text{ and } i \in \{0,1,2\}, \text{and} \\
\mathrm{me}(c) &= \emptyset, \ \text{for all other classes } c \in \mathcal{C}.
\end{aligned}
$$

Note that the constructed POB-schema S is pseudo-consistent. It finally remains to show that the undirected graph (V, E) is 3-colorable iff S is consistent.

($\Rightarrow$) If (V, E) is 3-colorable, then a mapping $\gamma : V \to \{0, 1, 2\}$ exists with $\gamma(u) \neq \gamma(v)$ for all $\{u, v\} \in E$. As easily checked, the following interpretation ε, where $\mathcal{O} = \{o_0, o_1, o_2, o_3\}$, is a model of S:

$$\begin{aligned}
\varepsilon(b) &= \{o_0, o_1, o_2, o_3\}\\
\varepsilon(c_1) &= \{o_3\}\\
\varepsilon(c_2) &= \{o_0, o_1, o_2\}\\
\varepsilon(d) &= \{o_3\}\\
\varepsilon(c_v^i) &= \{o_j, o_3\}, \text{ where } j = (\gamma(v) + i) \bmod 3 \quad (\text{for all } v \in V \text{ and } i \in \{0,1,2\})\\
\varepsilon(d_v^i) &= \{o_j\}, \text{ where } j = (\gamma(v) + i) \bmod 3 \quad (\text{for all } v \in V \text{ and } i \in \{0,1,2\})\\
\varepsilon(d_{u,v}^i) &= \{o_j\}, \text{ where } j = (\gamma(v) + i) \bmod 3 \quad (\text{for all } d_{u,v}^i \in \mathcal{C} \text{ where } i \in \{0,1,2\}).
\end{aligned}$$

($\Leftarrow$) If ε is a model of S, then it satisfies $\varepsilon(c_2) \neq \emptyset$ and the following conditions:

$$\begin{aligned}
&|\varepsilon(d_v^i)| = 1/3 \cdot |\varepsilon(c_2)| \text{ for all } v \in V \text{ and } i \in \{0,1,2\}\\
&\varepsilon(d_v^i) \cap \varepsilon(d_v^j) = \emptyset, \text{ for all } v \in V \text{ and } i, j \in \{0,1,2\} \text{ with } i \neq j\\
&\varepsilon(d_{u,v}^i) \cap \varepsilon(d_{v,u}^i) = \emptyset, \text{ for all } \{u, v\} \in E \text{ and } i \in \{0,1,2\}\\
&\varepsilon(d_v^1) \cup \varepsilon(d_v^2) \cup \varepsilon(d_v^3) = \varepsilon(c_2), \text{ for all } v \in V\\
&\varepsilon(d_v^i) = \varepsilon(d_{v,u}^i), \text{ for all } d_{u,v}^i \in \mathcal{C} \text{ where } i \in \{0,1,2\} \text{ (as } d_v^i \Rightarrow c_v^i,\ d_{v,u}^i \Rightarrow c_v^i).
\end{aligned}$$

Since $\varepsilon(c_2)$ is nonempty, there exists some $o \in \varepsilon(c_2)$. Let the mapping $\gamma : V \to \{0,1,2\}$ be defined by $\gamma(v) = i$ iff $o \in \varepsilon(d_v^i)$ for all nodes $v \in V$. It is now easy to see that γ is well-defined and that $\gamma(u) \neq \gamma(v)$ for all edges $\{u, v\} \in E$. Hence, (V, E) is 3-colorable. □

We next concentrate on the proof of Theorem 5.3. We need the following two lemmata.

Lemma 11.1 *Let* $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ *be a well-structured POB-schema. Let* $c \in \mathcal{C}$ *be any class and let* $\mathcal{P}_1, \mathcal{P}_2 \in \mathrm{me}(c)$ *be two distinct partition clusters. Then the common subclasses of* $\mathcal{P}_1$ *and* $\mathcal{P}_2$ *that are maximal with respect to* $\Rightarrow^\star$ *are pairwise t-disjoint.*

Proof. The claim follows immediately from W2. □

Lemma 11.2 *Let* $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ *be a well-structured POB-schema. Let* $c \in \mathcal{C}$ *such that* $G_{\mathbf{S}}(c)$ *contains at least one edge. Let* $\mathcal{P}_l$ *be a leaf or isolated node in a graph* $G = (\mathcal{V}, \mathcal{E})$ *that results from* $G_{\mathbf{S}}(c)$ *by iteratively removing leaves or isolated nodes. Suppose* $d_l \in \mathcal{C}$ *is a subclass of* $\mathcal{P}_l$ *but not of any other node* $\mathcal{P}$ *in* G*, and that* $d \in \mathcal{C}$ *is a subclass of some node* $\mathcal{P}$ *in* G *but not of* $\mathcal{P}_l$*. Then* d_l *and* d *are not t-disjoint and* $d \notin \mathrm{S}^\star(d_l)$*,* $d_l \notin S^\star(d)$*.*

Proof. We give a proof by contradiction. Assume first that d_l and d are t-disjoint. This means that there must be some class $c' \in \mathcal{C}$, some partition cluster $\mathcal{P}' \in \mathrm{me}(c')$, and two distinct classes $d_l', d' \in \mathcal{P}'$ such that $d_l \Rightarrow^\star d_l'$ and $d \Rightarrow^\star d'$. Suppose now that c' is not a subclass of c. By W1, there exists a top element $c_\top$ with $d_l' \Rightarrow^\star c_\top$ and $d' \Rightarrow^\star c_\top$. By W4, all paths from d_l up to $c_\top$ via d_l' and c' and all paths from d up to $c_\top$ via d' and c' must go through c. Thus, c must be a subclass of both d_l' and d'. But this contradicts P2. Hence, c' must be a subclass of c. Suppose now that $c = c'$. Thus, $\mathcal{P}'$

belongs to $\mathrm{me}(c)$. By the assumptions on d_l and d, the cluster $\mathcal{P}'$ cannot belong to G, which means it was removed in the construction of G from $\mathbf{s}(c)$. However, since the edges $\{\mathcal{P}_l, \mathcal{P}'\}$ and $\{\mathcal{P}, \mathcal{P}'\}$ are in $G_{\mathbf{S}}(c)$, this raises a contradiction. It follows that c' must be a proper subclass of c and thus a subclass of some cluster $\mathcal{P}'' \in \mathrm{me}(c)$. Again, by the assumptions on d_l and d, this cluster $\mathcal{P}''$ cannot belong to G, and assuming that $\mathcal{P}''$ was in the construction of G raises again a contradiction. This proves that d_l and d are t-disjoint.

Assume next that $d_l \in \mathrm{S}^\star(d)$ holds. Then d_l is a subclass of $\mathcal{P}$, which contradicts that d_l is not a subclass of any node in G different from $\mathcal{P}_l$. The case $d \in \mathrm{S}^\star(d)$ is analogous. □

We are now ready to prove Theorem 5.3.

Proof of Theorem 5.3. Let $\mathrm{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ be a pseudo-consistent and well-structured POB-schema. Let $c_\top$ denote the top element of S. By P1 and W1, there exists a unique mapping $\alpha : \mathcal{C} \to [0, 1]$ such that $\alpha(c_\top) = 1$ and that $\alpha(c) = \wp(c, d) \cdot \alpha(d)$ for all $c, d \in \mathcal{C}$ with $c \Rightarrow d$ (define $\alpha(c)$ as the product of the edge probabilities on any path from c to $c_\top$).

It is now easy to see that S is consistent iff there exists a finite nonempty set $\mathcal{O}$, a mapping ε from $\mathcal{C}$ to $2^{\mathcal{O}}$, and a mapping μ from $\mathcal{O}$ to the set of all rational numbers in $[0, 1]$ such that:

(1) $\varepsilon(c) \subseteq \varepsilon(d)$ for all classes $c, d \in \mathcal{C}$ with $c \Rightarrow d$.

(2) $\varepsilon(c) \cap \varepsilon(d) = \emptyset$ for all distinct classes $c, d \in \mathcal{C}$ that belong to the same $\mathcal{P} \in \bigcup\{\mathrm{me}(e) \mid e \in \mathcal{C}\}$.

(3) $\sum_{o \in \mathcal{O}} \mu(o) \ = \ 1$.

(4) $\alpha(c) = \sum_{o \in \varepsilon(c)} \mu(o)$ for all classes $c \in \mathcal{C}$.

Informally, μ assigns a mass to each object (possibly 0), and $\alpha(c)$ is the share of the total mass of all objects in c. Such $\mathcal{O}$, ε, and μ are now constructed by induction on the structure of S. Owing to possibly complex relationships between classes in S, the proof is naturally involved.

We first consider the case where S has no multiple inheritance, i.e., $(\mathcal{C}, \Rightarrow)$ is a tree with root $c_\top$.

Basis: For $\mathcal{C} = \{c_\top\}$, we define $\mathcal{O} = \{o\}$, $\varepsilon(c_\top) = \{o\}$, and $\mu(o) = 1$. It is easy to see that these $\mathcal{O}$, ε, and μ have the properties (1)–(4) in S.

Induction: Let $\mathcal{P}_1, \ldots, \mathcal{P}_l$ be the partition clusters in $\mathrm{me}(c_\top)$. Let $\mathbf{S}_1^\star = (\mathcal{C}_1^\star, \sigma_1^\star, \Rightarrow_1^\star, \mathrm{me}_1^\star, \wp_1^\star), \ldots,$ $\mathbf{S}_l^\star = (\mathcal{C}_l^\star, \sigma_l^\star, \Rightarrow_l^\star, \mathrm{me}_l^\star, \wp_l^\star)$ be the greatest POB-schemas that are contained in S, that have the top element $c_\top$, and that do not contain any classes from $\mathrm{S}(c_\top) - \mathcal{P}_1, \ldots, \mathrm{S}(c_\top) - \mathcal{P}_l$.

Without loss of generality, let us consider the partition cluster $\mathcal{P}_1 = \{c_1, \ldots, c_k\}$. Let $\mathbf{S}_i = (\mathcal{C}_i, \sigma_i, \Rightarrow_i, \mathrm{me}_i, \wp_i)$ be the greatest POB-schema that is contained in S and has the top element c_i, $forall i \in \{1, \ldots, k\}$. Each $\mathbf{S}_i$ is pseudo-consistent and well-structured. Hence, by the induction hypothesis, there exist $\mathcal{O}_i$, ε_i, and μ_i that satisfy (1)–(4) in $\mathbf{S}_i$. It is now easy to verify that the following $\mathcal{O}_1^\star$, $\varepsilon_1^\star$, and $\mu_1^\star$ have the properties (1)–(4) in $\mathbf{S}_1^\star$:

- $\mathcal{O}_1^\star = \{o\} \cup \bigcup_{i=1}^{k} \{c_i\} \times \varepsilon(c_i)$.
- $\varepsilon_1^\star(c_\top) = \mathcal{O}_1^\star$ and $\varepsilon_1^\star(d_i) = \{c_i\} \times \varepsilon_i(d_i)$ for all $d_i \in \mathcal{C}_i$ and $i \in \{1, \ldots, k\}$.
- $\mu_1^\star(o) = 1 - \sum_{i=1}^{k} \wp(c_i, c_\top)$ and $\mu_1^\star(c_i, o_i) = \wp(c_i, c_\top) \cdot \mu_i(o_i)$, for all $i \in \{1, \ldots, k\}, o_i \in \mathcal{O}_i$.

That is, there exist $\mathcal{O}_1^\star, \ldots, \mathcal{O}_l^\star, \varepsilon_1^\star, \ldots, \varepsilon_l^\star$, and $\mu_1^\star, \ldots, \mu_l^\star$ that satisfy (1)–(4) in $\mathbf{S}_1^\star, \ldots, \mathbf{S}_l^\star$. We finally define $\mathcal{O}$, ε, and μ with the properties (1)–(4) in $\mathbf{S}$ as follows:

- $\mathcal{O} = \mathcal{O}_1^\star \times \cdots \times \mathcal{O}_l^\star$.
- $\varepsilon(c_\top) = \mathcal{O}$ and $\varepsilon(d_i^\star) = \{(o_1^\star, \ldots, o_l^\star) \in \mathcal{O} \mid o_i^\star \in \varepsilon_i^\star(d_i^\star)\}$ for all $d_i^\star \in \mathcal{C}_i^\star - \{c_\top\}$ and $i \in \{1, \ldots, l\}$.
- $\mu(o_1^\star, \ldots, o_l^\star) = \prod_{i=1}^{l} \mu_i^\star(o_i^\star)$ for all $(o_1^\star, \ldots, o_l^\star) \in \mathcal{O}$.

Let us next concentrate on POB-schemas $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ with limited multiple inheritance: For every $c \in \mathcal{C}$ and every two distinct $\mathcal{P}_1, \mathcal{P}_2 \in \mathrm{me}(c)$, it holds that every $c_1 \in \mathcal{P}_1$ and $c_2 \in \mathcal{P}_2$ have at most one common subclass (which must then be different from c_1, c_2).

In this case, we must be more careful in the induction step. Along the same line of argumentation, we may construct $\mathcal{O}_i^\star$, $\varepsilon_i^\star$, and $\mu_i^\star$ with (1)–(4) in $\mathbf{S}_i^\star$, for all $i \in \{1, \ldots, l\}$, since P2 ensures that any two classes in the same partition cluster have pairwise disjoint sets of subclasses. The construction of $\mathcal{O}$, ε, and μ, however, is by induction on the structure of $G_{\mathbf{S}}(c_\top)$ as follows.

Basis: For $l = 1$, we define $\mathcal{O} = \mathcal{O}_1^\star$, $\varepsilon = \varepsilon_1^\star$, and $\mu = \mu_1^\star$.

Induction: For $l > 1$ and $G_{\mathbf{S}}(c_\top) = (\mathrm{me}(c_\top), \emptyset)$ (no common subclasses), the proof is as above.

For $l > 1$ and $G_{\mathbf{S}}(c_\top) \neq (\mathrm{me}(c_\top), \emptyset)$, by W3, the undirected graph $G_{\mathbf{S}}(c_\top)$ contains at least one leaf or one isolated node. Without loss of generality, let this node be given by $\mathcal{P}_l$. Let $\mathbf{S}' = (\mathcal{C}', \sigma', \Rightarrow', \mathrm{me}', \wp')$ be the POB-schema that unifies the POB-schemas $\mathbf{S}_1^\star, \ldots, \mathbf{S}_{l-1}^\star$ under the top element $c_\top$. By the induction hypothesis, there exist $\mathcal{O}'$, ε', and μ' with the properties (1)–(4) in $\mathbf{S}'$.

Let $e_1, \ldots, e_m$ with $m \geq 0$ denote all the classes in $\mathcal{C}' \cap \mathcal{C}_l^\star - \{c_\top\}$. It is now important to point out that, by the assumption on common subclasses and Lemma 11.1, the classes e_i are pairwise t-disjoint. Moreover, by Lemma 11.2, each classes $c' \in \mathcal{C}' - \{c_\top, e_1, \ldots, e_m\}$ and $c^\star \in \mathcal{C}^\star - \{c_\top, e_1, \ldots, e_m\}$ are not t-disjoint and $c' \notin \mathrm{S}^\star(c^\star)$, $c^\star \notin \mathrm{S}^\star(c')$ hold. Let $\widehat{\mathcal{O}} = \mathcal{O}' - (\varepsilon'(e_1) \cup \cdots \cup \varepsilon'(e_m))$ and $\widehat{\mathcal{O}}_l^\star = \mathcal{O}_l^\star - (\varepsilon_l^\star(e_1) \cup \cdots \cup \varepsilon_l^\star(e_m))$. We now define $\mathcal{O}$, ε, and μ as follows:

- $\mathcal{O} = \widehat{\mathcal{O}}' \times \widehat{\mathcal{O}}_l^\star \cup \bigcup_{j=1}^{m} \varepsilon'(e_j) \times \varepsilon_l^\star(e_j)$.

- $\varepsilon(d) = \begin{cases} \mathcal{O} & \text{if } d = c_\top \\ \varepsilon'(d) \times \varepsilon^\star_l(d) & \text{if } d \in \{e_1, \ldots, e_m\} \\ \{(o', o^\star_l) \in \mathcal{O} \mid o' \in \varepsilon'(d)\} & \text{if } d \in \mathcal{C}' - \{c_\top, e_1, \ldots, e_m\} \\ \{(o', o^\star_l) \in \mathcal{O} \mid o^\star_l \in \varepsilon^\star_l(d)\} & \text{if } d \in \mathcal{C}^\star - \{c_\top, e_1, \ldots, e_m\}\,. \end{cases}$

- $\mu(o', o^\star_l) = \begin{cases} \mu'(o') \cdot \mu^\star_l(o^\star_l) \,/\, \alpha(e_j) & \text{if } (o', o^\star_l) \in \varepsilon'(e_j) \times \varepsilon^\star_l(e_j) \\ \mu'(o') \cdot \mu^\star_l(o^\star_l) \,/\, (1 - \sum_{j=1}^{m} \alpha(e_j)) & \text{if } (o', o^\star_l) \in \widehat{\mathcal{O}}' \times \widehat{\mathcal{O}}^\star_l\,. \end{cases}$

(Note that $\sum_{o' \in \varepsilon'_l(e_j)} \mu'(o') = \sum_{o^\star_l \in \varepsilon^\star_l(e_j)} \mu^\star_l(o^\star_l) = \alpha(e_j)$ and $\sum_{o' \in \widehat{\mathcal{O}}'} \mu'(o') = \sum_{o^\star_l \in \widehat{\mathcal{O}}^\star_l} \mu^\star_l(o^\star_l) = 1 - \sum_{j=1}^{m} \alpha(e_j)$.) It can now easily be shown that $\mathcal{O}$, ε, and μ have the properties (1)–(4) in $\mathbf{S}$.

Let us finally consider the case of pseudo-consistent and well-structured POB-schemas $\mathbf{S}$ without any further restrictions. In this case, the classes $e_1, \ldots, e_m$ of the previous induction may have proper subclasses. The main ideas of the proof are now as follows. By the induction hypothesis, the subschema $\mathbf{S}_{e_j}$ under e_j is consistent. Since the classes e_j are pairwise disjoint, and each path from a class in $\mathbf{S}_{e_j}$ beyond that schema must pass through e_j, we can informally assume that the subgraph below e_j is replaced by a partition cluster $\{e_{j,1}, \ldots, e_{j,n_j}\}$ with $\sum_{i \in \{1,\ldots,n_j\}} \wp(e_{j,i}, e_j) = 1$ for all $j \in \{1, \ldots, m\}$, where we introduce a class $e_{j,i}$ for each object in the model of $\mathbf{S}_{e_j}$. We then apply the same construction as above, using $e_{j,1}, \ldots, e_{j,n_j}$, instead of e_j, for all $j \in \{1, \ldots, m\}$. □

Proof of Theorem 7.1. Immediate from the fact that the consistency of $\mathbf{S}$ is independent of σ. □

Proof of Theorem 7.2. Immediate from the fact that the consistency of $\mathbf{S}$ is independent of σ. □

Proof of Theorem 7.4. Analogous to Theorem 7.3 (observe that the consistency of a POB-schema is independent of the type assignment). □

Proof of Theorem 8.1. Let $\mathbf{I}_2 = (\pi_2, \nu_2) = \sigma_{\phi_2}(\mathbf{I})$, $\mathbf{I}_{1,2} = (\pi_{1,2}, \nu_{1,2}) = \sigma_{\phi_1}(\sigma_{\phi_2}(\mathbf{I}))$, and $\mathbf{I}_{1\wedge 2} = (\pi_{1\wedge 2}, \nu_{1\wedge 2}) = \sigma_{\phi_1 \wedge \phi_2}(\mathbf{I})$. Since $\phi_1 \wedge \phi_2$ is logically equivalent to $\phi_2 \wedge \phi_1$, it suffices to show that $\mathbf{I}_{1,2}$ and $\mathbf{I}_{1\wedge 2}$ coincide. For each $c \in \mathcal{C}$, we have

$$\begin{aligned} \pi_{1,2}(c) &= \{o \in \pi_2(c) \mid \mathrm{prob}_{\mathbf{I}_2,o} \models \phi_1\} \\ &= \{o \in \pi(c) \mid (\mathrm{prob}_{\mathbf{I},o} \models \phi_2) \wedge (\mathrm{prob}_{\mathbf{I}_2,o} \models \phi_1)\} \\ &= \{o \in \pi(c) \mid (\mathrm{prob}_{\mathbf{I},o} \models \phi_2) \wedge (\mathrm{prob}_{\mathbf{I},o} \models \phi_1)\} \\ &= \{o \in \pi(c) \mid \mathrm{prob}_{\mathbf{I},o} \models \phi_1 \wedge \phi_2\} \\ &= \pi_{1\wedge 2}(c); \end{aligned}$$

the key fact is that $\mathrm{prob}_{\mathbf{I}_2,o} \models \phi_1$ is equivalent to $\mathrm{prob}_{\mathbf{I},o} \models \phi_1$, if o belongs to $\pi_2(c)$, i.e., $\mathrm{prob}_{\mathbf{I},o} \models \phi_2$ is true. Furthermore, for each class $c \in \mathcal{C}$,

$$\begin{aligned}
\nu_{1,2}(c) &= \nu_2(c)|\pi_{1,2}(c) \\
&= \nu(c)|\pi_2(c)|\pi_{1,2}(c) \\
&= \nu(c)|\pi_{1,2}(c) && \text{as } \pi_{1,2}(c) \subseteq \pi_2(c) \\
&= \nu(c)|\pi_{1\wedge 2}(c) && \text{as } \pi_{1,2}(c) = \pi_{1\wedge 2}(c) \\
&= \nu_{1\wedge 2}(c).
\end{aligned}$$

This proves the result. □

Proof of Theorem 8.2. Equation (2): immediate from the fact that $\Pi_{\mathbf{A}}(\Pi_{\mathbf{B}}(\mathbf{I})) = \Pi_{\mathbf{A}\cap\mathbf{B}}(\mathbf{I})$.

Equation (3): By the definition of selection, the oid o is selected from $\mathbf{I}$ iff $\mathrm{prob}_{\,\mathbf{I},o} \models \phi$ holds. Since ϕ accesses only attributes in the schema $\mathbf{S}' = \Pi_{\mathbf{A}}(\mathbf{S})$, and since the projection $\Pi_{\mathbf{A}}(\mathbf{I}) =: \mathbf{I}'$ lets all such components of o survive, we have that $\mathrm{prob}_{\,\mathbf{I},o} \models \phi$ iff $\mathrm{prob}_{\,\mathbf{I}',o} \models \phi$, for every oid o. Thus, by definition of $\Pi_{\mathbf{A}}$ it follows that $\Pi_{\mathbf{A}}(\sigma_\phi(\mathbf{I})) = \Pi_{\mathbf{A}}(\sigma_\phi(\mathbf{I}'))$. Since $\Pi_{\mathbf{A}}(\sigma_\phi(\mathbf{I}'))$ and $\sigma_\phi(\mathbf{I}')$ have the same schema, we have $\Pi_{\mathbf{A}}(\sigma_\phi(\mathbf{I}')) = \sigma_\phi(\mathbf{I}'))$. It follows $\Pi_{\mathbf{A}}(\sigma_\phi(\mathbf{I})) = \sigma_\phi(\Pi_{\mathbf{A}}(\mathbf{I}))$. □

Proof of Theorem 8.3. Equation (4): Let $\mathbf{I} = (\pi, \nu)$. Consider any top-level attribute E from the schema of $\delta_N(\mathbf{I})$. Then, there is a top-level attribute D from $\mathbf{S}$ (not necessarily different from E) such that $\delta_N(D) = E$. Let o be any oid, and set $\phi' := \delta_{N^{-1}}(\phi)$. We then obtain:

$$\delta_N(\nu(o)).E = \delta_N(\nu(o).D) = \delta_N(\nu(o).\delta_{N^{-1}}(E)).$$

By induction on the structure of ϕ, we thus obtain that $\mathrm{prob}_{\,\delta_N(\mathbf{I}),o} \models \phi$ iff $\mathrm{prob}_{\,\mathbf{I},o} \models \phi'$ holds for all oids o populating $\mathbf{I}$ (equivalently, $\delta_N(\mathbf{I})$). Thus, o belongs to $\sigma_\phi(\delta_N(\mathbf{I}))$ iff it belongs to $\delta_N(\sigma_{\phi'}(\mathbf{I}))$, and since selection and renaming do not change class membership, o belongs in $\sigma_\phi(\delta_N(\mathbf{I}))$ and in $\delta_N(\sigma_{\phi'}(\mathbf{I}))$ to the same class. Since selection does not affect the value assignment, o has in $\sigma_\phi(\delta_N(\mathbf{I}))$ and in $\delta_N(\sigma_{\phi'}(\mathbf{I}))$ also the same probabilistic value. Thus $\sigma_\phi(\delta_N(\mathbf{I})) = \delta_N(\sigma_{\phi'}(\mathbf{I}))$.

Equation (5): The proof is analogous as for classical relational databases (class hierarchy and probability assignment do not play a role here).

We first observe that the schemas on the two sides coincide. Indeed, if A is a top-level attribute of the schema of $\Pi_{\mathbf{A}}(\delta_N(\mathbf{I}))$, then $A \in \mathbf{A}$ and $\delta_{N^{-1}}(A)$ is a top-level attribute of $\mathbf{S}$. Thus, $\delta_N(\delta_{N^{-1}}(A)) = A$ is a top-level attribute of the schema of $\delta_N(\Pi_{\delta_{N^{-1}}(\mathbf{A})}(\mathbf{I}))$. Conversely, if A is a top-level attribute of $\delta_N(\Pi_{\delta_{N^{-1}}(\mathbf{A})}(\mathbf{I}))$, then $\delta_{N^{-1}}(A)$ is a top-level attribute of $\mathbf{S}$ and, moreover, $A \in \mathbf{A}$. Thus, A is a top-level attribute of the schema of $\Pi_{\mathbf{A}}(\delta_N(\mathbf{I}))$. It follows from this that the schemes on the two sides of Equation (5) coincide. Since projection and renaming do not affect class membership of objects, $\mathbf{I}$, $\Pi_{\mathbf{A}}(\delta_N(\mathbf{I}))$, and $\delta_N(\Pi_{\delta_{N^{-1}}(\mathbf{A})}(\mathbf{I}))$ are populated by the same objects o, with the same class membership function. The value of the (existing) attribute A of object o in both $\Pi_{\mathbf{A}}(\delta_N(\mathbf{I}))$ and $\delta_N(\Pi_{\delta_{N^{-1}}(\mathbf{A})}(\mathbf{I}))$ is given by $\nu(o).\delta_{N^{-1}}(A)$, where $\mathbf{I} = (\pi, \nu)$. This establishes that $\Pi_{\mathbf{A}}(\delta_N(\mathbf{I}))$

and $\delta_N(\Pi_{\delta_{N-1}(\mathbf{A})}(\mathbf{I}))$ coincide. $\Box$

Proof of Theorem 8.4. Equation (6) (commutativity of the join) holds because (i) the join in classical databases (used to generate the POB-schema of the join and the oids of its instance) is commutative, (ii) classical set intersection $\cap$ is commutative, and (iii) the conjunction strategy $\otimes$ is commutative.

Equation (7) (associativity of join): A simple analysis of the possible cases for membership of an attribute A in the sets of top-level attributes $\mathbf{A}_1$, $\mathbf{A}_2$, and $\mathbf{A}_3$ of $\mathbf{S}_1, \mathbf{S}_2$, and $\mathbf{S}_3$, respectively, establishes that the join of probabilistic tuple values is associative. From associativity of classical relational join and set intersection, it is then easy to see from the definition of join for POB-instances that $(\mathbf{I}_1 \bowtie_\otimes \mathbf{I}_2) \bowtie_\otimes \mathbf{I}_3 = \mathbf{I}_1 \bowtie_\otimes (\mathbf{I}_2 \bowtie_\otimes \mathbf{I}_3)$ holds.

Equation (8): By Theorem 8.1 we have $\sigma_{\phi_1 \wedge \phi_2 \wedge \phi_3}(\mathbf{I}_1 \bowtie_\otimes \mathbf{I}_2) = \sigma_{\phi_3}(\sigma_{\phi_1 \wedge \phi_2}(\mathbf{I}_1 \bowtie_\otimes \mathbf{I}_2))$. Thus, it remains to establish that $\sigma_{\phi_1 \wedge \phi_2}(\mathbf{I}_1 \bowtie_\otimes \mathbf{I}_2) = \sigma_{\phi_1}(\mathbf{I}_1) \bowtie_\otimes \sigma_{\phi_2}(\mathbf{I}_2)$ holds.

The set of top-level attributes of the schema of $\mathbf{I}_{1 \bowtie 2} := \mathbf{I}_1 \bowtie_\otimes \mathbf{I}_2$ is $\mathbf{A}_1 \cup \mathbf{A}_2$. For each $A \in \mathbf{A}_1 - \mathbf{A}_2$, its value for an object $o = (o_1, o_2)$ in $\mathbf{I}_{1 \bowtie 2}$, provided A exists for o, is by definition of join given by $\nu_1(o_1).A$. Since ϕ_1 involves only top-level attributes from $\mathbf{A}_1 - \mathbf{A}_2$, this implies $\mathrm{prob}_{\mathbf{I}_{1\bowtie 2}, o} \models \phi_1$ iff $\mathrm{prob}_{\mathbf{I}_1, o_1} \models \phi_1$. Similarly, we obtain that $\mathrm{prob}_{\mathbf{I}_{1\bowtie 2}, o} \models \phi_2$ iff $\mathrm{prob}_{\mathbf{I}_2, o_2} \models \phi_2$; hence, $\mathrm{prob}_{\mathbf{I}_{1\bowtie 2}, o} \models \phi_1 \wedge \phi_2$ iff $\mathrm{prob}_{\mathbf{I}_i, o_i} \models \phi_i$, for $i \in \{1, 2\}$.

Thus, for every oid $o = (o_1, o_2)$ belonging to $\sigma_{\phi_1 \wedge \phi_2}(\mathbf{I}_1 \bowtie_\otimes \mathbf{I}_2)$ it holds that o_i is in $\sigma_{\phi_i}(\mathbf{I}_i)$, for $i \in \{1, 2\}$, and thus (o_1, o_2) belongs to $\sigma_{\phi_1}(\mathbf{I}_1) \bowtie_\otimes \sigma_{\phi_2}(\mathbf{I}_2)$. Moreover, since selection does not change an object's value assignment, o has in $\sigma_{\phi_1}(\mathbf{I}_1) \bowtie_\otimes \sigma_{\phi_2}(\mathbf{I}_2)$ the same value as in $\sigma_{\phi_1 \wedge \phi_2}(\mathbf{I}_{1\bowtie 2})$. Conversely, if $o = (o_1, o_2)$ belongs to $\sigma_{\phi_1}(\mathbf{I}_1) \bowtie_\otimes \sigma_{\phi_2}(\mathbf{I}_2)$, then o belongs to $\mathbf{I}_{1\bowtie 2}$, and $\mathrm{prob}_{\mathbf{I}_{1\bowtie 2}, o} \models \phi_1 \wedge \phi_2$ holds. Thus, o belongs also to $\sigma_{\phi_1 \wedge \phi_2}(\mathbf{I}_{1\bowtie 2})$, and has in it the same value as in $\sigma_{\phi_1}(\mathbf{I}_1) \bowtie_\otimes \sigma_{\phi_2}(\mathbf{I}_2)$. This proves $\sigma_{\phi_1 \wedge \phi_2}(\mathbf{I}_1 \bowtie_\otimes \mathbf{I}_2) = \sigma_{\phi_1}(\mathbf{I}_1) \bowtie_\otimes \sigma_{\phi_2}(\mathbf{I}_2)$.

Equation (9): Since projection does not change class membership, both $\Pi_{\mathbf{B}}(\mathbf{I}_1 \bowtie_\otimes \mathbf{I}_2)$ and $\Pi_{\mathbf{B}}(\Pi_{\mathbf{B}_1}(\mathbf{I}_1) \bowtie_\otimes \Pi_{\mathbf{B}_2}(\mathbf{I}_2))$ are populated by the objects $o = (o_1, o_2)$ from $\mathbf{I}_1 \bowtie_\otimes \mathbf{I}_2$. The inner projections $\Pi_{\mathbf{B}_1}$, $\Pi_{\mathbf{B}_2}$ remove all top-level attributes $A \in \mathbf{A}_1 \cup \mathbf{A}_2$ from o_1 and o_2, respectively, which are neither common attributes of $\mathbf{A}_1$ and $\mathbf{A}_2$ nor in $\mathbf{B}$. Each such attribute of o in $\mathbf{I}_1 \bowtie_\otimes \mathbf{I}_2$ is removed by the projection $\Pi_{\mathbf{B}}$ applied to it. Thus, the value of o in $\Pi_{\mathbf{B}}(\mathbf{I}_1 \bowtie_\otimes \mathbf{I}_2)$ and $\Pi_{\mathbf{B}_1}(\mathbf{I}_1) \bowtie_\otimes \Pi_{\mathbf{B}_2}(\mathbf{I}_2)$ is the same. It follows $\Pi_{\mathbf{B}}(\mathbf{I}_1 \bowtie_\otimes \mathbf{I}_2) = \Pi_{\mathbf{B}}(\Pi_{\mathbf{B}_1}(\mathbf{I}_1) \bowtie_\otimes \Pi_{\mathbf{B}_2}(\mathbf{I}_2))$. $\Box$

Proof of Theorem 8.6. Equations (14) and (15) are immediate from commutativity and associativity, respectively, of set intersection and the probabilistic conjunction strategy $\otimes$.

Equations (16) and (17) can be concluded similarly, with disjunction strategy $\oplus$ in place of $\otimes$, based on the following observation: without loss of generality, we may assume that each object o occurring in one of $\mathbf{I}_1$, $\mathbf{I}_2$, and $\mathbf{I}_3$ occurs in all of them and belongs to the same class; furthermore, the set V of values of a top-level attribute A associated with o is always the same. The reason is that technically, we may add o to the POB-instances in which it is missing, and add values v to V in each $\mathbf{I}_i$ and set $\alpha(v) = \beta(v) = 0$ such that V is everywhere the same. Thanks to the postulate of Ignorance for $\oplus$, this does not change the result of the expressions in Equations (16) and (17). Under this technical assumption, only the intersection case in the definitions of $[\alpha(v), \beta(v)]$ (Def. 7.27) and $\nu(o)$ as in Def. 7.29 is relevant, for which the properties obviously hold.

Equation (18) holds since projection simply removes attributes while it does not affect class membership of objects, and the intersection $pt_1 \cap_\otimes pt_2$ of probabilistic tuple values pt_1 and pt_2 is the aggregation of (independent) intersections $pt_1.A_i \otimes pt_2.A_i$ of all their attributes A_i. The argument for Equations (19) and (20) is analogous (as above, we may add oids o and values v to simplify the value assignment to o in $\mathbf{I}_1 \cup_\oplus \mathbf{I}_2$ and $\mathbf{I}_1 -_\ominus \mathbf{I}_2$, respectively). $\Box$