Temporal Probabilistic Object Bases

Veronica Biazzo* Rosalba Giugno* Thomas Lukasiewicz† V.S. Subrahmanian‡

Abstract

There are numerous applications where we have to deal with temporal uncertainty associated with objects. The ability to automatically store and manipulate time, probabilities, and objects is important. We propose a data model and algebra for temporal probabilistic object bases (TPOBs) which allows us to specify the probability with which an event occurs at a given time point. In *explicit* TPOB-instances, the sets of time points along with their probability intervals are explicitly enumerated. In *implicit* TPOB-instances, sets of time points are expressed by constraints and their probability intervals by probability distribution functions. Thus, implicit object base instances are succinct representations of explicit ones; they allow for an efficient implementation of algebraic operations, while their explicit counterparts make defining algebraic operations easy. We extend the relational algebra to both explicit and implicit instances and prove that the operations on implicit instances correctly implement their counterpart on explicit instances.

Index terms: probabilistic databases, uncertainty management, temporal data.

1 Introduction

Object data models [1, 5, 37, 4] have been used to model multimedia applications [11, 12], financial risk applications [10], logistics and supply chain management applications [2], weather applications [14] as well as many others. Many of these applications naturally need to represent and manipulate both time and uncertainty.

• Commercial shipping companies (such as UPS, Fedex, DHL, etc.) have detailed statistics on how long packages take to get from one zip code to another, and often even more specific information (e.g., how long it takes for a package from one street address to another). A company

*Dipartimento di Matematica e Informatica, Università di Catania, Viale A. Doria 6, I-95125 Catania, Italy; e-mail: {vbiazzo, giugno}@dmi.unict.it.

†Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Via Salaria 113, I-00198 Rome, Italy; e-mail: lukasiewicz@dis.uniroma1.it. Alternate address: Institut für Informationssysteme, Technische Universität Wien, Favoritenstraße 9-11, A-1040 Vienna, Austria; e-mail: lukasiewicz@kr.tuwien.ac.at.

‡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.

expecting deliveries would like to have some information about when the deliveries will arrive of the form "the package will be delivered between 9 am and 1pm with a probability between 0.1 and 0.2, and between 1pm and 5pm with a probability between 0.8 and 0.9". Such an answer is far more helpful to corporate decision making than saying "It will be delivered sometime today between 9 am and 5pm". This helps them schedule personnel, prepare receiving facilities (for hazardous or bulky materials), prepare future production plans, etc. In addition, the different entities involved in such an application are typically stored using object models — this is because different vehicles (airplanes, trucks, etc.) have different capabilities, and because different packages (letter, tube, hazardous material shipments for commercial customers, etc.) have widely varying attributes. The shipping company itself has extensive need for such data. For example, the company would need to query this database to create plans that optimally allocate and/or use available resources (space on trucks, personnel, etc.) based on their expectations of the probable workload at future time points.

• In past work, we have built a surveillance system (jointly with TASC Inc. and DARPA) where image recognition programs are used to identify enemy vehicles types, as well as when and where they were spotted. Recently, this has been extended in a major demonstration (given in Nov. 2002) involving over 20 corporate and academic partners to the US Department of Defense where such information is used to predict where the enemy vehicles may be in the future. Here, two kinds of uncertainty are present. The first is uncertainty about the type of enemy entity - is it a tracked vehicle or a wheeled vehicle? Tracked vehicles include tanks and armored personnel carriers (APCs). Wheeled vehicles include cars, trucks, etc. Likewise, radars that may be spotted include thin-skinned and thick-skinned radars. Tanks may include T-66, T-72, and T-80 tanks. For a given enemy, estimates of the number of their tanks and APCs are known (and similarly for thin-skinned versus thick-skinned radars), allowing probabilities of a tracked vehicle being a tank or an APC to be generated. Likewise, based on such numbers, we can infer the probability of an arbitrary enemy tank being a T-72 tank. The second kind of probability is generated by the predictive models used: when (if ever) will the detected enemy entity reach a given location on the ground? Uncertainty arises here because of lack of information about the enemy vehicle, its speed, its mission, and how weather and terrain conditions (as well as battlefield conditions) will affect its progress. Here, the uncertainty ranges over time.

• Institutional lenders and banks use financial models [3, 10] to predict when customers will default on credit. Such models are complex mathematical models involving probabilities and time (the predictions specify the probability with which a given customer will default over a given period of time). Furthermore, models to predict bankruptcies and loan defaults vary substantially depending

upon the market, the type of credit instrument (consumer credit card, mortgage, commercial real estate loan, HUD loan, construction loan, etc.), the variables that affect the loan, various aspects about the customer, etc. Such models are naturally represented via object models, and time and uncertainty parameterize various features of the model.

This paper proposes a theoretical foundation for object bases that handles probabilistic temporal indeterminacy. We illustrate the use of our model on a simplified transportation logistics example which runs through this paper. The main contributions of the paper are:

• We define the concepts of a TPOB-schema, and (explicit) TPOB-instances over a TPOB-schema. Such TPOB-instances represent a probabilistic statement over a set of time points by simply enumerating all time points along with their probability intervals.
• We define algebraic operations that operate on TPOB-instances. We define selection, restricted selection, projection, extraction (a new operation), natural join and Cartesian product. In [6], we also define renaming, intersection, union, difference and condition join. Projection and Cartesian product are extensions of their counterparts from the relational algebra, while all other operations are full-fledged complex-object operations (as they address the "inner" components of object values).
• We introduce the notion of consistency for TPOB-instances and a related notion called coherence and show that under appropriate assumptions, all our algebraic operations preserve consistency and coherence of schemas and instances, respectively.
• Given an event e, for each time point t, a TPOB-instance $\mathbf{I}$ *explicitly* specifies a probability interval[1] describing the probability with which event e occurred (or will occur) at time t. Explicit TPOB-instances make it easy to formally define the various algebraic operations - in addition, they can make it easy for end users to understand the result. However, they can be very expensive in terms of memory (and hence also inefficient from the point of view of computing algebraic operators). For example, if a company wants to record that a particular shipment will arrive sometime between 8am and 5pm on day D, and the temporal granularity used is seconds, then we need to make this statement for each of the $9 \times 60 \times 60 = 32400$ seconds between 8am and 5pm. *Implicit*

[1] As in the case of [32, 19, 20], our model uses interval probabilities rather than point probabilities. The reasons are three-fold: First, probability intervals are a generalization of point probabilities, and thus can handle all point probabilities. Second, they allow expression of imprecise probabilistic knowledge. For example, it is possible to say that an event occurred at time t with probability p with a margin of error δ, denoting that the probability of e occurring at time t lies in the interval $[p-\delta, p+\delta]$. Hence, probability intervals are better suited especially to represent statistical knowledge and subjective degrees of belief (cf. [31]). Third, even if we know a point probability for two events e_1 and e_2 it is well known that in general, **we can only infer a probability interval** (cf. Boole [8], Fagin et al. [21]) for the conjunction and disjunction of e_1 and e_2. For example, if we know that the probability of e_1 is 0.6 and the probability of e_2 is 0.8, in general, we can only infer the probability of $(e_1 \wedge e_2)$ to lie in the range $[0.4, 1]$. If additional information is known, Boole [8], the probability is guaranteed to fall into this interval.

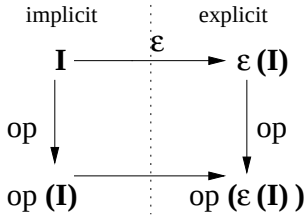

Figure 1: Correctness theorems: commutativity diagram

TPOB-instances avoid this problem.

• We define a translation ε that maps implicit TPOB-instances **I** to explicit ones, $\varepsilon(\mathbf{I})$. For each algebraic operation on explicit TPOB-instances, we define a counterpart on implicit TPOB-instances.

The **key advantage** of the explicit representation is that it is easy to understand from a user's point of view, and that defining the algebraic operations is not tricky. The **key advantage** of the implicit representation is that it is space efficient as well as time efficient (as the operations work on a succinct representation of the data).

• We show that the algebraic operations on implicit TPOB-instances correctly implement their counterparts on explicit TPOB-instances . Figure 1 provides a diagrammatic representation of what these results look like for unary operators (a similar figure can be shown for binary operators).

The rest of this paper is organized as follows. Section 2 provides an overview of related work. In Section 3, we introduce some basic definitions. Section 4 presents the notions of TPOB-schemas and of explicit TPOB-instances. In Sections 5 and 6, we introduce the unary and binary, respectively, algebraic operations on explicit TPOB-instances. In Section 7, we show that they preserve coherence and consistency of schemas and instances. Sections 8 and 9 define implicit TPOB-instances and the algebraic operations on implicit TPOB-instances, respectively. Section 10 contains directions for future work. For convenience, a summary of notation used in the paper is given in Table 13 at the end of the paper. Note that detailed proofs of all results are given in [6].

2 Related Work

Though there has been extensive work on temporal databases and temporal object oriented databases [35, 24, 44, 41], there has been less work on probabilistic object bases [26, 27, 20, 19] and to date, no work at all that we are aware of on temporal probabilistic data in object bases. This paper is a temporal extension of the model by Eiter et al. [19].

Dyreson and Snodgrass [18] extend *relational* DBs and *SQL* with probabilistic uncertainty on time points. They add indeterminate temporal attributes (which have indeterminate instants as asso-

ciated values) to SQL. Indeterminate instants are intervals of time points with associated probability distributions. SQL is extended by the *ordering plausibility* construct. This is an integer between 1 and 100. 1 means that any possible answer to a query is desired, while 100 says that only definite answers to a query are desired). Numbers in between reflect interim positions. The *correlation credibility* construct modifies probability distributions in the base relations before evaluating the selection condition in SQL queries. Dyreson and Snodgrass [18] also describe efficient data structures and query processing algorithms for their approach. Dekhtyar et al. [13] extend the *relational data model and algebra* with probabilistic temporal indeterminacy. They define a *theoretical annotated temporal algebra* on large *annotated relations*, and a *temporal probabilistic algebra* on succinct *temporal probabilistic relations*. They show that the latter efficiently and correctly implements the former. They also report on timings of the temporal probabilistic algebra in a prototype implementation.

Nonprobabilistic temporal indeterminacy in databases has been studied by Snodgrass [40] who models indeterminacy using a model that is based on a three-valued logic. Dutta [17] and Dubois and Prade [16] propose a fuzzy logic approach to temporal indeterminacy, while Koubarakis [29, 28] and Brusoni et al. [9] suggest approaches based on constraints. Gadia et al. [22] introduce partial temporal databases, which are based on partial temporal elements.

Our work is closest in spirit to the above work by Dekhtyar et al. [13]. The idea of having an explicit algebra on large instances, which is efficiently and correctly implemented by an implicit algebra on succinct instances, is inspired by their work. Our work is an extension of the much richer object-oriented data model and algebra, as compared to the relational algebra. Thus, our work may be viewed as a generalization of the one by Dekhtyar et al. [13]. In contrast to Dyreson and Snodgrass [18], we present an extension of object-oriented DBs, while their approach is an extension of relational DBs. Second, we make no independence assumptions between events (the user's query can explicitly encode her knowledge of the dependencies between events, if any), while Dyreson and Snodgrass assume that all indeterminate events are mutually independent. Third, our work introduces an algebra, while their work defines an SQL extension. Fourth, we present formal definitions of important notions like coherence and consistency and show that under appropriate assumptions, our operations all preserve coherence and consistency. Fifth, we allow for interval probabilities over solution sets of temporal constraints, while they only allow for precise point probabilities over intervals of time points.

Our work is also related to data models and algebraic operations for complex objects [1, 38, 45, 43, 42, 7] and is a strict extension of the complex value algebra of Abiteboul et al. [1]. Like Shaw and Zdonik [38], Vandenberg and DeWitt [45], and Boncz et al. [7], our data model sup-

ports type constructors for sets and tuples on elementary datatypes. Like them, we also support selection, projection, join, union, intersection, and difference. However, unlike them [45], we do not support type constructors for arrays and multisets, user-defined abstract datatypes [38], classes as elementary datatypes [7], and aggregate and nest/unnest operations. Our work involves time and probabilities that are not considered in these papers. The nested relational algebra described in [43] is a functional language for complex objects, which also allows for defining the high-level algebraic operations of selection, projection, Cartesian product, intersection, and difference [43]. Finally, Subramanian et al. [42] describe an object-oriented query algebra for lists and trees. They also present a predicate language for lists and trees, which supports order-sensitive queries, as it is based on pattern matching. Such algebraic operations are related to our extraction operation, which extracts a subhierarchy from the class hierarchy of a temporal object base.

3 Basic Definitions

In this section, we define calendars [30], types, and values of a type. We then define probabilistic types and their values. Finally, we describe the concept of a probabilistic strategy which is used to combine probabilistic information in our algebraic operations in Sections 5 and 6.

Calendars. Intuitively, a calendar consists of a finite sequence of time units and a predicate specifying a set of valid time points over this sequence.

A *time unit* $T = (N, V)$ consists of a *name* N and a set of *time values* V. We often use N to refer to T. A *linear temporal hierarchy* $H = T_1 \sqsupseteq \ldots \sqsupseteq T_n$ consists of a finite set of distinct time units $\{T_1, \ldots, T_n\}$ and a linear order $\sqsupseteq$ among them. The following example illustrates the above concepts.

Example 3.1 $T_y = (year, \{0, 1, \ldots\})$, $T_m = (month, \{1, \ldots, 12\})$, and $T_d = (day, \{1, \ldots, 31\})$ are time units, while $T_y \sqsupseteq T_m \sqsupseteq T_d$, or $year \sqsupseteq month \sqsupseteq day$, is a linear temporal hierarchy. □

A linear temporal hierarchy specifies a set of time points, while a calendar additionally specifies a subset of valid time points. More formally, a *time point* over $H = T_1 \sqsupseteq \ldots \sqsupseteq T_n$ is a tuple $(t_1, \ldots, t_n)$, where each t_i is a time value of T_i. We denote by $<_H$ the lexicographic order on all time points over H, which is defined by: $(s_1, \ldots, s_n) <_H (t_1, \ldots, t_n)$ iff some $i \in \{1, \ldots, n\}$ exists such that $s_j = t_j$ for all $j \in \{1, \ldots, i-1\}$ and $s_i < t_i$. We use $\leq_H$ to denote the reflexive closure of $<_H$. A *calendar* $C = (H, P)$ consists of a linear temporal hierarchy H and a *validity predicate* P, which specifies a nonempty set of *valid* time points over H. A calendar is *finite* if the set of all its valid time points is finite. In the rest of this paper, all calendars are finite unless specified otherwise. The reader interested in how to specify validity predicates may consult [30]. We give an

example to illustrate the concepts of time points and calendars.

Example 3.2 $(1997, 1, 31)$ and $(1997, 2, 31)$ are time points over $H = year \sqsupseteq month \sqsupseteq day$. In a calendar $C = (H, P)$, the validity predicate P may now characterize the former as valid and the latter as invalid. □

Classical Types. A calendar τ is a *temporal atomic type* whose domain dom(τ) consists of all the valid time points of τ. The set of *classical atomic types* is $\mathcal{T} = \{\mathsf{integer}, \mathsf{string}, \mathsf{Boolean}, \mathsf{float}\}$ with the usual domains. We also assume the existence of some arbitrary but fixed set $\mathcal{A}$ of attributes.

Classical types are either atomic types or complex types constructed from atomic types and attributes by using the set and the tuple constructor. Formally:

- Every classical atomic type from $\mathcal{T}$ and every temporal atomic type is a classical type.
- If τ is a classical type, then $\{\tau\}$ is a classical type (called *classical set type*).
- If $A_1, \ldots, A_k$ are pairwise distinct attributes from $\mathcal{A}$ and $\tau_1, \ldots, \tau_k$ are classical types, then $[A_1 : \tau_1, \ldots, A_k : \tau_k]$ is a classical type (called *classical tuple type*).

We give some examples of classical types.

Example 3.3 (Package Example) Transportation companies (e.g. UPS, DHL) have detailed data about how long it takes packages to get from one place to another. These companies store massive databases containing information about the various packages they are shipping or have previously shipped. Packages may be viewed as objects with a variety of attributes. The attributes Origin and Destination (defining where the package originated and is destined for) may be defined over the classical atomic type string. An attribute such as contents (Contents) may be defined over the classical set type {string}. A classical tuple type is [Origin: string, Destination: string, Contents: {string}]. *In the rest of this paper, we will continue to expand this Package Example.* □

The *values* of classical types are inductively defined as follows:

- For all classical atomic types $\tau \in \mathcal{T}$, every $v \in$ dom(τ) is a value of the classical type τ.
- If $v_1, \ldots, v_k$ are values of τ, then $\{v_1, \ldots, v_k\}$ is a value of the classical type $\{\tau\}$.
- If $A_1, \ldots, A_k$ are pairwise distinct attributes from $\mathcal{A}$ and $v_1, \ldots, v_k$ are values of $\tau_1, \ldots, \tau_k$, then $[A_1 : v_1, \ldots, A_k : v_k]$ is a value of the classical type $[A_1 : \tau_1, \ldots, A_k : \tau_k]$.

Some values of classical types in the Package Example are shown below.

Example 3.4 Boston is a value of the classical atomic type string, while {pens, books, camera} is a value of {string}. Moreover, [Origin: Boston, Destination: New_York, Contents: {pens, books, camera}] is a value of [Origin: string, Destination: string, Contents: {string}]. □

Observe that the above classical types can be easily extended to also include the type constructors for lists (i.e., ordered sets) and bags (i.e., multisets) in addition to the set and tuple constructors.

Probabilistic Types. Probabilistic types encode probabilistic information. They are either atomic probabilistic types, or complex probabilistic types constructed from classical types and atomic probabilistic types using the tuple constructor. Formally,

- If τ is a classical type, then $[\![\tau]\!]$ is a probabilistic type (called *atomic probabilistic type*).
- If $A_1, \ldots, A_k$ are pairwise distinct attributes from $\mathcal{A}$ and $\tau_1, \ldots, \tau_k$ are either classical or probabilistic types, then $[A_1 : \tau_1, \ldots, A_k : \tau_k]$ is a probabilistic type (called *probabilistic tuple type*). We call the attributes $A_1, \ldots, A_k$ its *top-level attributes*.

Note that ordinary (relational) tuple types are a special case of probabilistic types. Probabilistic set types may be defined in the obvious way — for space reasons, we omit them here.

Example 3.5 In the Package Example, the attributes Delivery and STOPone (giving delivery time and first stop during transportation, respectively) may be defined over the atomic probabilistic type $[\![\mathsf{time}]\!]$ and the probabilistic tuple type $[\mathsf{City}: \mathsf{string}, \mathsf{Arrive}: \mathsf{string}, \mathsf{Shipment}: [\![\mathsf{time}]\!]]$, respectively, where time is a calendar. □

The values of probabilistic types are appropriately typed random variables. Formally,

- A value of an atomic probabilistic type $[\![\tau]\!]$ is a finite set of pairs $(v, [l, u])$, where v is a value of τ, and l, u are reals with $0 \leq l \leq u \leq 1$.
- A value of a probabilistic type $[A_1: \tau_1, \ldots, A_k: \tau_k]$ is of the form $v = [A_1: v_1, \ldots, A_k: v_k]$, where $v_1, \ldots, v_k$ are values of $\tau_1, \ldots, \tau_k$. We write $v.A_i$ to denote v_i.

Intuitively, a probabilistic value $v = \{(v_1, [l_1, u_1]), \ldots, (v_n, [l_n, u_n])\}$ says that v's value is exactly one member of the set $\{v_1, \ldots, v_n\}$ and that the probability that v's value is v_i lies in the interval $[l_i, u_i]$. This is illustrated below.

Example 3.6 Let time be the calendar over the linear temporal hierarchy $hour \sqsupseteq minute$. A value v of type $[\![\mathsf{time}]\!]$ is $\{((12, 30), [.4, .6]), ((12, 35), [.4, .6])\}$. This says that value v is either $(12, 30)$ or $(12, 35)$. The probability that the value is $(12, 30)$ is 0.4–0.6 and similarly for $(12, 35)$. □

For a probabilistic value $v = \{(v_1, [l_1, u_1]), \ldots, (v_n, [l_n, u_n])\}$ to be consistent, there must be some way of assigning point probabilities to the v_i's so that the above constraints are satisfied. For example, $v = \{(5, [1, 1]), (6, [1, 1])\}$ is inconsistent (as it says that v equals both 5 and 6 with probability 1). v is *consistent* iff there exists a probability function Pr (cf. [36]) on $\{v_1, \ldots, v_n\}$ (that is, a mapping $Pr: \{v_1, \ldots, v_n\} \to [0, 1]$ such that all $Pr(v_i)$ sum up to 1) such that $Pr(v_i) \in [l_i, u_i]$ for all $i \in \{1, \ldots, n\}$. It is easy to see that such a Pr exists iff $l_1 + \cdots + l_n \leq 1 \leq u_1 + \cdots + u_n$. Hence,

v is *consistent* iff $v_1, \ldots, v_n$ are pairwise distinct and $l_1 + \cdots + l_n \leq 1 \leq u_1 + \cdots + u_n$. A value v of a probabilistic type is *consistent* iff all values of atomic probabilistic types that occur in v are consistent.

Probabilistic Strategies. Consider two events e_1 and e_2 having a probability in the intervals $[l_1, u_1]$ and $[l_2, u_2]$, respectively. To compute the probability interval associated with the compound events $e_1 \vee e_2$ and $e_1 \wedge e_2$, we need to know the dependencies between e_1 and e_2 (or lack thereof). For instance, e_1 and e_2 may be mutually exclusive, or probabilistically independent, or positively correlated (when e_1 implies e_2, or e_2 implies e_1), or we may be ignorant of the relationship between e_1 and e_2. Each of these situations yields a different way of computing the probability of $e_1 \vee e_2$ and $e_1 \wedge e_2$. Formally, let $\mathcal{U}$ be the set of all nonempty subintervals $[l, u]$ of the unit interval $[0, 1]$. Suppose e_1 and e_2 have probabilities in the intervals $[l_1, u_1]$ and $[l_2, u_2]$, respectively. A *conjunction* (resp., *disjunction*) *strategy* is a function $\otimes \colon \mathcal{U}^2 \to \mathcal{U}$ (resp., $\oplus \colon \mathcal{U}^2 \to \mathcal{U}$) that computes the probability interval of $e_1 \wedge e_2$ (resp., $e_1 \vee e_2$) for some fixed dependencies between e_1 and e_2 (or lack thereof). The table below shows some examples of conjunction and disjunction strategies.

Strategy	Conjunction	Disjunction
Mutual exclusion	[0,0]	$[\min(1, l_1 + l_2), \min(1, u_1 + u_2)]$
Positive correlation	$[\min(l_1, l_2), \min(u_1, u_2)]$	$[\max(l_1, l_2), \max(u_1, u_2)]$
Independence	$[l_1 \cdot l_2,\ u_1 \cdot u_2]$	$[l_1 + l_2 - l_1 \cdot l_2,\ u_1 + u_2 - u_1 \cdot u_2]$
Ignorance	$[\max(0, l_1 + l_2 - 1), \min(u_1, u_2)]$	$[\max(l_1, l_2), \min(1, u_1 + u_2)]$

In our framework, users can specify (in their query) the relevant strategy they wish to use. Each of our algebraic operations (in both algebras presented in this paper) is parameterized by a strategy.

4 Temporal Probabilistic Object Bases

A TPOB-schema (introduced below) consists of two parts — a hierarchy of classes with associated types and a conditional probability. If c_1 is an immediate subclass of c_2, this conditional probability specifies the probability of a member of c_2 being a member of c_1. For example, c_1 could be the class "registered_letters" and c_2 could be the class "letters." Or it could be the probability of a tracked vehicle being a tank (or a given tank being a T-72). Second, we define the inheritance completion of a schema, which adds to every class, all the attributes (with their types) that are inherited from superclasses. Finally, we introduce TPOB-instances with respect to the inheritance completion of a schema.

4.1 Temporal Probabilistic Object Base Schema

A schema contains a finite set of classes $\mathcal{C}$. Every class $c \in \mathcal{C}$ has an associated probabilistic tuple type $\sigma(c)$ specifying the type of objects in this class. Moreover, every $c \in \mathcal{C}$ is associated with a partition $\mathrm{me}(c)$ of the set of all its immediate subclasses into sets (or clusters) of pairwise disjoint classes. For example, $\mathrm{me}(c) = \{\{c_1, c_2\}, \{c_3, c_4, c_5\}\}$ says that $c_1, \ldots, c_5$ are the immediate subclasses of c, and that an object o that belongs to the class c can belong to at most one class among c_1 and c_2, and to at most one class among c_3, c_4, and c_5. Finally, every immediate subclass relationship between two classes c_1 and c_2 is associated with the conditional probability that an arbitrary object belongs to c_1 given that it belongs to c_2.

Formally, a *temporal probabilistic object base schema* (or *TPOB-schema*) $\mathbf{S} = (\mathcal{C}, \sigma, \mathrm{me}, \wp)$ consists of (i) a finite set of *classes* $\mathcal{C}$, (ii) a *type assignment* σ that associates with each class $c \in \mathcal{C}$ a probabilistic tuple type, (iii) a mapping me that associates with each class $c \in \mathcal{C}$ a partition of the set of all its *immediate subclasses* $d \in \mathcal{C}$, and (iv) a *probability assignment* $\wp$ that associates with every pair of classes $(c_i, c) \in \mathcal{C} \times \mathcal{C}$, where c_i is an immediate subclass of c, a positive rational number in $[0, 1]$ such that $\Sigma_{c_i \in \mathcal{P}} \wp(c_i, c) \leq 1$ for every class $c \in \mathcal{C}$ and every cluster $\mathcal{P} \in \mathrm{me}(c)$. Observe that me defines a directed graph $(\mathcal{C}, \Rightarrow)$, where $c_1 \Rightarrow c_2$ iff c_1 is an immediate subclass of c_2. As usual, we assume that $(\mathcal{C}, \Rightarrow)$ is acyclic. The following example illustrates the concept of a TPOB-schema.

Example 4.1 The TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \mathrm{me}, \wp)$ for the Package Example is given as follows:

- $\mathcal{C} = \{\mathsf{Package}, \mathsf{Letter}, \mathsf{Box}, \mathsf{Tube}, \mathsf{Priority}, \mathsf{Express\_saves}, \mathsf{One\text{-}transfer}, \mathsf{Two\text{-}transfer}\}$;
- the type assignment σ is given by Table 1 below;
- $\mathrm{me}(\mathsf{Package}) = \{\{\mathsf{Letter}, \mathsf{Box}, \mathsf{Tube}\}, \{\mathsf{Priority}, \mathsf{Express\_saves}\}\}$,
  $\mathrm{me}(\mathsf{Box}) = \mathrm{me}(\mathsf{Express\_saves}) = \{\{\mathsf{Two\text{-}transfer}\}\}$,
  $\mathrm{me}(\mathsf{Tube}) = \mathrm{me}(\mathsf{Priority}) = \{\{\mathsf{One\text{-}transfer}\}\}$,
  $\mathrm{me}(\mathsf{Letter}) = \mathrm{me}(\mathsf{One\text{-}transfer}) = \mathrm{me}(\mathsf{Two\text{-}transfer}) = \emptyset$;
- the probability assignment $\wp$ is given in Fig. 2.

The directed acyclic graph $(\mathcal{C}, \Rightarrow)$ is obtained from the graph of Fig. 2 by contracting the d-nodes to Package and removing probability labels. The d-nodes of Fig. 2 denote disjoint decompositions — every package is either a letter, box, or tube, and either an express or a priority package. □

A *top-level attribute* of a TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \mathrm{me}, \wp)$ is a top-level attribute (as defined in Section 3) of some $\sigma(c)$ where $c \in \mathcal{C}$. A *directed path* in $(\mathcal{C}, \Rightarrow)$ is a sequence of classes $c_1, \ldots, c_k \in \mathcal{C}$ such that $c_1 \Rightarrow \cdots \Rightarrow c_k$ and $k \geq 1$. $\Rightarrow^\star$ denotes the reflexive, transitive closure of $\Rightarrow$. c_1 is a *subclass* (resp., *strict subclass*) of c_2, or c_2 is a *superclass* (resp., *strict superclass*) of

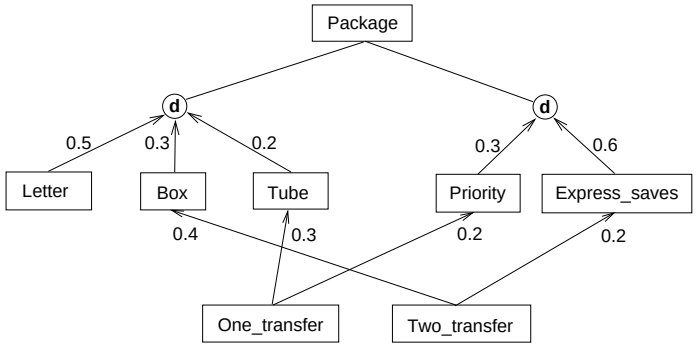

Figure 2: Package Example with probability assignment $\wp$

Table 1: Type assignment σ

c	$\sigma(c)$
Package	[Origin: string, Destination: string, Delivery: [[time]]]
Letter	[Height: float, Width: float]
Box	[Height: float, Width: float, Depth: float, Contents: {string}]
Tube	
Priority	[Time: [[time]]]
Express_saves	
One-transfer	[City: string, Arrive: [[time]], Shipment: [[time]]]
Two-transfer	[STOPone: [City: string, Arrive: [[time]], Shipment: [[time]]], STOPtwo: [City: string, Arrive: [[time]], Shipment: [[time]]]]

c_1, iff $c_1 \Rightarrow^\star c_2$ (resp., $c_1 \Rightarrow^\star c_2$ and $c_1 \neq c_2$). A class $d \in \mathcal{C}$ is *minimal* under $\Rightarrow^\star$ in a set of classes $\mathcal{D} \subseteq \mathcal{C}$ iff $d \in \mathcal{D}$ and no class in $\mathcal{D}$ is a strict subclass of d.

Intuitively, a TPOB-schema is consistent iff each class c can be associated with a nonempty set $\zeta(c)$ of objects such that the immediate subclass and disjointness relationships and the relative cardinalities expressed by me and $\wp$, respectively, are satisfied. If there is no way to make this happen, then this means that there is some fundamental problem with the conditional probability assignments to the TPOB-schema. Formally, an *interpretation* $\mathcal{I} = (\mathcal{O}, \zeta)$ of a TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \mathrm{me}, \wp)$ consists of a nonempty set $\mathcal{O}$, and a mapping ζ from $\mathcal{C}$ to the set of all finite subsets of $\mathcal{O}$. $\mathcal{I}$ is a *taxonomic model* of $\mathbf{S}$ iff:

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

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

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

The first two axioms say that classes must not be empty and that the objects in a subclass must be a subset of the objects in a superclass. To see why the third axiom is present, consider the fact (from Fig. 2) that a package is either a tube, box, or letter — this axiom forces a package to be exactly one of these three (e.g., a package cannot be both a tube and a box).

$\mathcal{I}$ is a *taxonomic and probabilistic model* (or *model*) of $\mathbf{S}$ iff it is a taxonomic model of $\mathbf{S}$ and for all classes $c, d \in \mathcal{C}$ with $c \Rightarrow d$, $|\zeta(c)| = \wp(c,d) \cdot |\zeta(d)|$. $\mathbf{S}$ is *consistent* iff a model of $\mathbf{S}$ exists. We say that two classes $c, d \in \mathcal{C}$ are *taxonomically disjoint* (or *t-disjoint*) iff $\zeta(c) \cap \zeta(d) = \emptyset$ for every taxonomic model $\mathcal{I} = (\mathcal{O}, \zeta)$ of $\mathbf{S}$.

The above definition says that the number of items in each class must be consistent with the conditional probability labeling. For example, if an interpretation assigns 100 objects to the class "Package", then it must assign 50 objects to the class "Letter" — if not, it will not satisfy the probability requirement that half the packages are letters. More details about consistency may be found in [6].

4.2 Inheritance Completion

The inheritance completion of a TPOB-schema adds to the type of every class $c \in \mathcal{C}$, all attributes (with their types) inherited from all superclasses of c. When a class can inherit an attribute from multiple superclasses, we find all minimal such superclasses and use an *inheritance strategy* to choose from which of these "minimally above" superclasses the property will be inherited.

Formally, let $\mathbf{S} = (\mathcal{C}, \sigma, \mathrm{me}, \wp)$ be a TPOB-schema, and let $\boldsymbol{A}$ denote the set of all its top-level attributes. Let $\mathsf{min}_{\mathbf{S}} : \mathcal{C} \times \boldsymbol{A} \to \mathcal{C}$ be the mapping that assigns to each pair $(c, A) \in \mathcal{C} \times \boldsymbol{A}$ the set of all minimal classes under $\Rightarrow^\star$ in the set of all superclasses $d \in \mathcal{C}$ of c such that A is a top-level attribute of $\sigma(d)$. An *inheritance strategy* for $\mathbf{S}$ is a partial mapping $\mathsf{inh}_{\mathbf{S}} : \mathcal{C} \times \boldsymbol{A} \to \mathcal{C}$ that assigns a class $d \in \mathsf{min}_{\mathbf{S}}(c, A)$ to every pair $(c, A) \in \mathcal{C} \times \boldsymbol{A}$ such that $\mathsf{min}_{\mathbf{S}}(c, A) \neq \emptyset$.

The inheritance completion of a TPOB-schema is obtained by adding to each type of a class, all top-level attributes with their types that are inherited from minimal superclasses. More formally, the *inheritance completion TPOB-schema* of a TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \mathrm{me}, \wp)$ is the TPOB-schema $\mathbf{S}^\star = (\mathcal{C}, \sigma^\star, \mathrm{me}, \wp)$, where $\sigma^\star(c) = [A_1 : \tau_1, \ldots, A_k : \tau_k]$ such that (i) $A_1, \ldots, A_k$ are all $A \in \mathcal{A}$ such that $\mathsf{inh}_{\mathbf{S}}(c, A)$ is defined, and (ii) $\sigma(\mathsf{inh}_{\mathbf{S}}(c, A_i))$ has the type τ_i at A_i, for all $i \in \{1, \ldots, k\}$. If $\mathbf{S} = \mathbf{S}^\star$, then $\mathbf{S}$ is *fully inherited*. The following example shows the inheritance completion of the TPOB-schema in our Package Example.

Example 4.2 Consider the TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \mathrm{me}, \wp)$ of Example 4.1. Its inheritance completion TPOB schema is $\mathbf{S}^\star = (\mathcal{C}, \sigma^\star, \mathrm{me}, \wp)$, where $\sigma^\star$ is given in Table 2. □

Throughout the rest of this paper, we assume that all TPOB-schemas are fully inherited.

Table 2: Type assignment $\sigma^\star$ of the inheritance completion TPOB-schema

c	$\sigma^\star(c)$
Package	[Origin: string, Destination: string, Delivery: [[time]]]
Letter	[Origin: string, Destination: string, Delivery: [[time]], Height: float, Width: float]
Box	[Origin: string, Destination: string, Delivery: [[time]], Height: float, Width: float, Depth: float, Contents: {string}]
Tube	
Priority	[Origin: string, Destination: string, Delivery: [[time]], Time: [[time]]]
Express_saves	
One-transfer	[Origin: string, Destination: string, Delivery: [[time]], Height: float, Width: float, Depth: float, Contents: {string}, Time: [[time]], City: string, Arrive: [[time]], Shipment: [[time]]]
Two-transfer	[Origin: string, Destination: string, Delivery: [[time]], Height: float, Width: float, Depth: float, Contents: {string}, Time: [[time]], STOPone: [City: string, Arrive: [[time]], Shipment: [[time]]], STOPtwo: [City: string, Arrive: [[time]], Shipment: [[time]]]]

4.3 Temporal Probabilistic Object Base Instance

This section introduces the notion of a temporal probabilistic object base (TPOB) instance. Throughout this paper, we assume a countably infinite set $\mathcal{O}$ of *object identifiers* (or *oids*) (we will often abuse notation and call *oids* objects). We assume $\mathcal{O}$ is countably infinite to ensure the existence of arbitrary large finite TPOB-instances. In the case of natural join and Cartesian product (see Section 6), we also assume that $\mathcal{O}$ is closed under Cartesian product, i.e., $\mathcal{O} \times \mathcal{O} \subseteq \mathcal{O}$.

A *temporal probabilistic object base instance* (or *TPOB-instance*) $\mathbf{I} = (\pi, \nu)$ over a consistent TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \mathrm{me}, \wp)$ consists of (i) a mapping π that assigns to each class $c \in \mathcal{C}$ a finite subset of $\mathcal{O}$ such that $\pi(c_1) \cap \pi(c_2) = \emptyset$ for all distinct classes $c_1, c_2 \in \mathcal{C}$, and (ii) a mapping ν that assigns to each oid $o \in \pi(c)$, $c \in \mathcal{C}$, a value of type $\sigma^\star(c)$. We write $\pi(\mathcal{C})$ to refer to the set $\bigcup\{\pi(c) \mid c \in \mathcal{C}\}$ of all oids in $\mathbf{I}$. For every $c \in \mathcal{C}$, we use $\pi^\star(c)$ to denote the set $\bigcup\{\pi(c') \mid c' \in \mathcal{C}, c' \Rightarrow^\star c\}$ of all oids that *belong* to c, which is to be distinguished from the set $\pi(c)$ of all oids that are *created* in c. Intuitively, $\pi(c)$ is the set of oid's in class c, while $\nu(o)$ specifies the value of an object $o \in \pi(c)$. The following example shows a TPOB-instance over the TPOB-schema in our Package Example.

Example 4.3 Tables 3 and 4 show a TPOB-instance $\mathbf{I}=(\pi, \nu)$ over the TPOB-schema $\mathbf{S}$ of Example 4.1. According to this, o_3 is a priority package, o_5 is a two-transfer package. o_3 is being shipped from Rome to Boston and its expected delivery time is either $(18, 00)$ or $(18, 31)$. The probability of the former is 40–60% while that of the latter is 30–50%. □

The probabilistic extent of a class specifies the probability with which a given object belongs to a class. For example, in the surveillance application mentioned in Section 1, there is a probability that an object seen on the battlefield is a T-72 tank. c is a mapping $\mathrm{ext(c)}$ that associates with every object o in the object base, the possible probabilities with which o belongs to c. If o

Table 3: The mappings π and π^*

c	$\pi(c)$	$\pi^*(c)$
Package	$\{\}$	$\{o_3, o_5\}$
Letter	$\{\}$	$\{\}$
Box	$\{\}$	$\{o_5\}$
Tube	$\{\}$	$\{\}$

c	$\pi(c)$	$\pi^*(c)$
Priority	$\{o_3\}$	$\{o_3\}$
Express_saves	$\{\}$	$\{o_5\}$
One-transfer	$\{\}$	$\{\}$
Two-transfer	$\{o_5\}$	$\{o_5\}$

Table 4: Value assignment ν

o	$\nu(o)$
o_3	[Origin: Rome, Destination: Boston, Delivery: $\{((18,00),[.4,.6]),((18,31),[.3,.5])\}$, Time: $\{((8,00),[.45,.5]),((8,10),[.4,.5])\}$]
o_5	[Origin: Paris, Destination: San_Jose, Delivery: $\{((12,00),[.5,.7]),((12,15),[.4,.5])\}$, Height: 60, Width: 50, Depth: 40, Contents: {photos, books}, Time: $\{((12,00),[.3,.4]),((12,05),[.4,.7])\}$, STOPone: [City: New_York, Arrive: $\{((14,00),[.3,.5]),((14,30),[.7,.8])\}$, Shipment: $\{((16,00),[1,1])\}$], STOPtwo: [City: ST_Louis, Arrive: $\{((17,30),[.2,.6]),((17,45),[.5,.6])\}$, Shipment: $\{((18,00),[.3,.5]),((18,30),[.6,.7])\}$]]

belongs to a subclass of c (resp., a class c' t-disjoint to c), then obviously $\mathrm{ext(c)}(o) = \{1\}$ (resp., $\mathrm{ext(c)}(o) = \{0\}$). Otherwise, $\mathrm{ext(c)}(o)$ is the set of all products of probabilities from c up to a minimal superclass c' of c that contains o. Formally, let $\mathbf{I} = (\pi, \nu)$ be a TPOB-instance over a consistent TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \mathrm{me}, \wp)$, and let c be a class from $\mathcal{C}$. 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, 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 minimal class c' under $\Rightarrow^\star$ in the set of all superclasses $d \in \mathcal{C}$ of c such that $o \in \pi^\star(d)\}$.

We call the class c in (1), the class c' in (2), and every class c' as in (3) a *characteristic class for* $\mathrm{ext(c)}(o)$. The following example illustrates the concept of a probabilistic extent of a class.

Example 4.4 The probabilistic extent of the class One-transfer in our Package Example maps the oids o_3 and o_5 to the sets of rational numbers $\{.2\}$ and $\{0\}$, respectively. □

If the probabilistic extent of every class c assigns a unique probability to every object o, then we say that the underlying TPOB-instance is coherent. Intuitively, there is only one probability that object o belongs to class c (otherwise something would be wrong). For example, we know that a package o in Figure 2, has a 0.06 probability of having one transfer (as Fig. 2 has two paths and the computed probability across *both* paths is 0.06). Formally, a TPOB-instance $\mathbf{I} = (\pi, \nu)$ over a consistent TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \mathrm{me}, \wp)$ is *coherent* iff for every class $c \in \mathcal{C}$ and every object $o \in \pi(\mathcal{C})$, the probabilistic extent $\mathrm{ext(c)}(o)$ contains *at most* one element. A TPOB-instance

$\mathbf{I} = (\pi, \nu)$ over a consistent TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \mathrm{me}, \wp)$ is *consistent* iff $\nu(o)$ is consistent for all $o \in \pi(\mathcal{C})$.

5 TPOB-Algebra: Unary Operations

In this section, we introduce selection, restricted selection, projection, and extraction. [6] also contains details about renaming and condition join. *Unless specified otherwise, we assume that all input TPOB-schemas are fully inherited.*

5.1 Selection

Selection finds all objects in a TPOB-instance $\mathbf{I}$ which satisfy a probabilistic selection condition ϕ. To define selection, we first define path expressions which are used to construct atomic selection conditions (ASCs). Selection conditions are Boolean combinations of ASCs. Probabilistic selection conditions (PSCs) are then defined. Intuitively, atomic PSCs say that a selection condition must be true with probability in some range. PSCs are Boolean combinations of atomic PSCs. Throughout this section, $\mathbf{I} = (\pi, \nu)$ is a TPOB-instance over a TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \mathrm{me}, \wp)$.

5.1.1 Path Expressions

Path expressions are finite sequences of attributes, which refer to "inner" attribute values of an object. Formally, a path expression for the tuple type $[A_1 \colon \tau_1, \ldots, A_k \colon \tau_k]$ has the form A_i or the form $A_i.P_i$, where P_i is a path expression for τ_i. A path expression for the atomic probabilistic type $[\![\tau]\!]$ is of the form $[\![P]\!]$, where P is a path expression for τ.

Example 5.1 STOPone, STOPone.City, and STOPone.Arrive are path expressions for the probabilistic tuple type [STOPone: [City: string, Arrive: [[time]], Shipment: [[time]]]]. □

We now define how a path expression P addresses a part of a value v. Given a path expression P for type τ, the *valuation* of P under a value v of τ, denoted $v.P$, is inductively defined by:

- if $v = [A_1 \colon v_1, \ldots, A_k \colon v_k]$ and $P = A_i$, then $v.P = v_i$;
- if $v = [A_1 \colon v_1, \ldots, A_k \colon v_k]$ and $P = A_i.R$, then $v.P = v_i.R$;
- if $v = \{(v_1, I_1), \ldots, (v_k, I_k)\}$ and $P = [\![R]\!]$, then $v.P = \{(v_1, I_1, R), \ldots, (v_k, I_k, R)\}$. We call such sets $\{(v_1, I_1, R), \ldots, (v_k, I_k, R)\}$ *generalized values* of τ.

Otherwise, $v.P$ is undefined. The following example illustrates this concept of valuation.

Example 5.2 The valuation of the path expression STOPone.City under the value [STOPone: [City: Boston, Arrive: $\{((8,00),[1,1])\}$, Shipment: $\{((9,00),[1,1])\}$]] is given by Boston. □

5.1.2 Atomic Selection Conditions

Atomic selection conditions are used to select objects satisfying some conditions. Path variables are used (instead of classical tuple variables) to specify conditions on attribute (or sub or sub-sub etc. attribute) values. Formally, an *atomic selection condition (ASC)* has one of the following forms:

- $in(c)$, where c is a class in $\mathcal{C}$;
- $P\,\theta\,v$, where P is a path expression, v is a value, and $\theta \in \{\leq, <, =, \neq, >, \geq\}$;
- $P_1\,\theta_{\otimes}\,P_2$, where P_1, P_2 are path expressions, $\otimes$ is a conjunction strategy, and $\theta \in \{\leq, <, =, \neq, >, \geq\}$.

Some examples of ASCs relating to the Package Example are given below.

Example 5.3 $in(\mathsf{Letter})$ is an ASC which specifies the set of all letters. Likewise, $\mathsf{STOPone.City}=\mathsf{New\_York}$ is an ASC which specifies the set of "all objects that have New York as the first stop." □

We now define the meaning of an ASC ϕ under an object o in $\mathbf{I}$. We associate with each such ϕ a closed subinterval of $[0, 1]$, which describes the range for the probability that the object o in $\mathbf{I}$ satisfies ϕ. Formally, the *probabilistic valuation* of ϕ with respect to $\mathbf{I}$ and $o \in \pi(\mathcal{C})$, denoted $\mathrm{prob}_{\mathbf{I},o}(\phi)$, is defined as follows (where $\oplus$ is the disjunction strategy for mutual exclusion):

- $\mathrm{prob}_{\mathbf{I},o}(in(c)) = [\min(\mathrm{ext(c)}(o)), \max(\mathrm{ext(c)}(o))]$.
- Let P be a path expression for the type of o. If $\nu(o).P$ is a value of a classical type, then define $V = \{(\nu(o), [1, 1], P)\}$, else if $\nu(o).P$ is a generalized value of an atomic probabilistic type, then define $V = \nu(o).P$. Otherwise, V is undefined. Then,

$$\mathrm{prob}_{\mathbf{I},o}(P\,\theta\,v) = \begin{cases} \bigoplus_{i=1}^{k} I_i & \text{if } V \text{ is defined} \\ \text{undefined} & \text{otherwise,} \end{cases}$$

  where $I_1, \ldots, I_k$ are the intervals I such that $(w, I, S) \in V$ and $w.S\,\theta\,v$, if V is defined. Note that $\mathrm{prob}_{\mathbf{I},o}(P\,\theta\,v)$ is undefined if some $w.S\,\theta\,v$ is undefined.

- For each $i \in \{1, 2\}$, let P_i be a path expression for the type of o. If $\nu(o).P_i$ is a value of a classical type, then define $V_i = \{(\nu(o), [1, 1], P_i)\}$, else if $\nu(o).P_i$ is a generalized value of an atomic probabilistic type, then define $V = \nu(o).P_i$. Otherwise, V_i is undefined. Then,

$$\mathrm{prob}_{\mathbf{I},o}(P_1\,\theta_{\otimes}\,P_2) = \begin{cases} \bigoplus_{i=1}^{k} I_i & \text{if } V_1 \text{ and } V_2 \text{ are defined} \\ \text{undefined} & \text{otherwise,} \end{cases}$$

where $I_1, \ldots, I_k$ are the intervals $I \otimes J$ such that $(v_1, I, S_1) \in V_1$, $(v_2, J, S_2) \in V_2$, and $v_1.S_1\, \theta\, v_2.S_2$, if V_1 and V_2 are defined. Note that $\mathrm{prob}_{\mathbf{I},o}(P_1\, \theta_{\otimes}\, P_2)$ is undefined, if some $v_1.S_1\, \theta\, v_2.S_2$ is undefined.

The following example shows some probabilistic valuations of atomic selection conditions.

Example 5.4 $in(\mathsf{One\text{-}transfer})$ is assigned $[.2, .2]$ (resp., $[0, 0]$) under $\mathrm{prob}_{\mathbf{I},o_3}$ (resp., $\mathrm{prob}_{\mathbf{I},o_5}$), while STOPone.City=New_York is undefined under $\mathrm{prob}_{\mathbf{I},o_3}$, and assigned $[1, 1]$ under $\mathrm{prob}_{\mathbf{I},o_5}$.

5.1.3 Selection Conditions

Selection conditions are logical combinations of atomic selection conditions. Formally: (i) Every ASC is a selection condition. (ii) If ϕ and ψ are selection conditions and $\otimes$ (resp., $\oplus$) is a conjunction (resp., disjunction) strategy, then $\phi \wedge_{\otimes} \psi$ (resp., $\phi \vee_{\oplus} \psi$) is a selection condition.

Example 5.5 $in(\mathsf{One\text{-}transfer}) \wedge_{\otimes_{in}} (\mathsf{Arrive} < (13, 00) \vee_{\oplus_{in}} \mathsf{Arrive} > (14, 00))$ is a selection condition which specifies "all objects in One-transfer that arrive before 13:00 or after 14:00". □

We now define the meaning of a selection condition ϕ under an object o in $\mathbf{I}$, by associating with each such ϕ a closed subinterval of $[0, 1]$, which describes the range for the probability that the object o in $\mathbf{I}$ satisfies ϕ. We do this by extending the probabilistic valuation $\mathrm{prob}_{\mathbf{I},o}$ as follows:

- $\mathrm{prob}_{\mathbf{I},o}(\phi \wedge_{\otimes} \psi) = \mathrm{prob}_{\mathbf{I},o}(\phi) \otimes \mathrm{prob}_{\mathbf{I},o}(\psi)$.
- $\mathrm{prob}_{\mathbf{I},o}(\phi \vee_{\oplus} \psi) = \mathrm{prob}_{\mathbf{I},o}(\phi) \oplus \mathrm{prob}_{\mathbf{I},o}(\psi)$.

Example 5.6 The selection condition $in(\mathsf{Priority}) \wedge_{\otimes_{in}} \mathsf{Time} < (14, 00)$ is assigned the probability intervals $[.95, 1]$ and $[0, 0]$ under $\mathrm{prob}_{\mathbf{I},o_3}$ and $\mathrm{prob}_{\mathbf{I},o_5}$, respectively. □

5.1.4 Probabilistic Selection Conditions

Probabilistic selection conditions (PSCs) are used to select all objects that satisfy an (ordinary) selection condition with a probability in a given range $[l, u]$. Formally, (i) If ϕ is a selection condition, and l, u are reals with $0 \leq l \leq u \leq 1$, then $(\phi)[l, u]$ is a an (*atomic*) PSC. (ii) If ϕ and ψ are PSCs, then so are $\neg\phi$ and $(\phi \wedge \psi)$. We use $(\phi \vee \psi)$ to abbreviate $\neg(\neg\phi \wedge \neg\psi)$. Some examples of probabilistic selection conditions for the Package Example are shown below.

Example 5.7 $(\mathsf{Time} < (14, 00))[.5, 1]$ is a probabilistic selection condition, which specifies "all objects whose value in the attribute Time is smaller than 14:00 with a probability in $[.5, 1]$". □

We now define what it means for an object o in $\mathbf{I}$ to satisfy a probabilistic selection condition ϕ. The *satisfaction* of ϕ under $\mathbf{I}$ and o, denoted $\mathrm{prob}_{\mathbf{I},o} \models \phi$, is inductively defined by:

- $\mathrm{prob}_{\mathbf{I},o} \models (\phi)[l, u]$ iff $\mathrm{prob}_{\mathbf{I},o}(\phi) \subseteq [l, u]$;

- $\text{prob}_{\mathbf{I},o} \models \neg\phi$ iff it is not the case that $\text{prob}_{\mathbf{I},o} \models \phi$;
- $\text{prob}_{\mathbf{I},o} \models \phi \wedge \psi$ iff $\text{prob}_{\mathbf{I},o} \models \phi$ and $\text{prob}_{\mathbf{I},o} \models \psi$.

The next example illustrates the satisfaction of probabilistic selection conditions.

Example 5.8 The following satisfaction (and non-satisfaction) relations hold in the Package Example:

- $\text{prob}_{\mathbf{I},o_3} \not\models (in(\mathsf{Priority}) \wedge_{\otimes_{in}} \mathsf{Time} < (14,00))[.96,1]$,
- $\text{prob}_{\mathbf{I},o_3} \models (in(\mathsf{Priority}) \wedge_{\otimes_{in}} \mathsf{Time} < (14,00))[.5,1]$,
- $\text{prob}_{\mathbf{I},o_3} \models (in(\mathsf{Priority}) \wedge_{\otimes_{in}} \mathsf{Time} < (14,00))[.5,1] \wedge (in(\mathsf{One\text{-}transfer}))[.2,.3]$. □

5.1.5 Selection Operation

The selection operator finds all objects in a probabilistic instance $\mathbf{I}$ that satisfy some probabilistic selection condition. Formally, the *selection* on $\mathbf{I}$ with respect to a probabilistic selection condition ϕ, denoted $\sigma_\phi(\mathbf{I})$, is the TPOB-instance (π', ν') over $\mathbf{S}$, where:

- $\pi'(c) = \{o \in \pi(c) \mid \models \text{is defined for } \text{prob}_{\mathbf{I},o} \text{ and } \phi, \text{ and } \text{prob}_{\mathbf{I},o} \models \phi\}$, for all $c \in \mathcal{C}$.
- $\nu' = \nu \mid \pi'(\mathcal{C})$.

The following example illustrates the use of the selection operator in the Package Example.

Example 5.9 Consider the TPOB-instance $\mathbf{I} = (\pi, \nu)$ of Example 4.3 and the probabilistic selection condition $\phi = \neg(in(\mathsf{Priority}) \wedge_{\otimes_{in}} \mathsf{Time} < (14,00))[0,0]$. Then, $\sigma_\phi(\mathbf{I}) = (\pi', \nu')$ contains the set of all objects in $\mathbf{I}$ that belong to Priority and have a value in Time smaller than 14:00 with a positive probability. Observe that $\pi'(\mathsf{Priority}) = \{o_3\}$, $\pi'(c) = \emptyset$ for all other classes c, and ν' is given by Table 5. □

Table 5: ν' resulting from selection

o	$\nu'(o)$
o_3	[Origin: Rome, Destination: Boston, Delivery: $\{((18,00),[.4,.6]),((18,31),[.3,.5])\}$, Time: $\{((8,00),[.45,.5]),((8,10),[.4,.5])\}$]

5.2 Restricted Selection

The selection operation selects a set of objects in a TPOB-instance without modifying them. However, there can be cases where we wish to do an ordinary selection on a probabilistic tuple type. For instance, if we wish to restrict the TPOB-instance in Example 5.9 to times before $(8,05)$, we

may just want to delete the information about $(8, 10)$ in Example 5.9. This is often needed so that the user can focus on the time points of interest to him or her without having to view probabilities about other time points.

We first formally define the restricted selection on values of probabilistic tuple types. Let τ be a probabilistic tuple type, and let $v = [A_1 : v_1, \ldots, A_i : v_i, \ldots, A_k : v_k]$ be a value of τ. Let ϕ be an ASC of the form $P \,\theta\, w$, where P is a path expression for τ. The *restricted selection* on v w.r.t. ϕ, denoted $\sigma^r_\phi(v)$, is defined as follows:

- If v_i is a value of a classical type, $P = A_i$, and $v_i \,\theta\, w$, then $\sigma^r_\phi(v) = v$.
- If v_i is a value of an atomic probabilistic type, and $P = A_i$, then $\sigma^r_\phi(v)$ is obtained from v by replacing v_i by $\{(w', I) \in v_i \mid w' \,\theta\, w\}$.
- If v_i is a value of a probabilistic tuple type, and $P = A_i.R$, then $\sigma^r_\phi(v)$ is obtained from v by replacing v_i by $\sigma^r_{R\,\theta\,w}(v_i)$.

Otherwise, $\sigma^r_\phi(v)$ is undefined.

We are now ready to extend the restricted selection operator to TPOB-instances as follows. Let $\mathbf{I}=(\pi, \nu)$ be a TPOB-instance over a TPOB schema $\mathbf{S} = (\mathcal{C}, \sigma, \mathrm{me}, \wp)$. Let ϕ be an atomic selection condition of the form $P \,\theta\, w$, where P is a path expression for every $\sigma(c)$ with $c \in \mathcal{C}$. The *restricted selection* on $\mathbf{I}$ with respect to ϕ, denoted $\sigma^r_\phi(\mathbf{I})$, is defined as the TPOB-instance (π', ν') over $\mathbf{S}$, where:

- $\pi'(c) = \{o \in \pi(c) \mid \sigma^r_\phi(\nu(o))$ is defined$\}$, for all $c \in \mathcal{C}$.
- $\nu'(o) = \sigma^r_\phi(\nu(o))$, for all $o \in \pi'(\mathcal{C})$.

5.3 Projection

Projection removes some top-level attributes (with their associated types) from a TPOB-schema, and the same attributes with their associated values from a TPOB-instance. Formally, the *projection* of a TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \mathrm{me}, \wp)$ on a set of top-level attributes $\mathbf{A}$ of $\mathbf{S}$, denoted $\Pi_{\mathbf{A}}(\mathbf{S})$, is the TPOB-schema $(\mathcal{C}, \sigma', \mathrm{me}, \wp)$, where the new type $\sigma'(c)$ of each class $c \in \mathcal{C}$ is obtained from the old type $\sigma(c) = [B_1 : \tau_1, \ldots, B_k : \tau_k]$ by removing all $B_i : \tau_i$'s with $B_i \notin \mathbf{A}$.

The *projection* of a TPOB-instance $\mathbf{I} = (\pi, \nu)$ over $\mathbf{S}$ on $\mathbf{A}$, denoted $\Pi_{\mathbf{A}}(\mathbf{I})$, is the TPOB-instance (π, ν') over the TPOB-schema $\Pi_{\mathbf{A}}(\mathbf{S})$, where the new value $\nu'(o)$ of each oid $o \in \pi(\mathcal{C})$ is obtained from the old value $\nu(o) = [B_1 : v_1, \ldots, B_k : v_k]$ by removing all $B_i : v_i$'s with $B_i \notin \mathbf{A}$.

The following example illustrates the projection of TPOB-schemas (resp., TPOB-instances).

Example 5.10 Consider the fully inherited TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \mathrm{me}, \wp)$ of Example 4.2, the TPOB instance $\mathbf{I} = (\pi, \nu)$ over $\mathbf{S}$ of Example 4.3, and the set of attributes $\mathbf{A} = \{\mathsf{Origin}, \mathsf{Contents},$

Time}. The projection of $\mathbf{S}$ (resp., $\mathbf{I}$) on $\mathbf{A}$ results in the TPOB-schema $\Pi_{\mathbf{A}}(\mathbf{S}) = (\mathcal{C}, \sigma', \mathrm{me}, \wp)$ (resp., TPOB-instance $\Pi_{\mathbf{A}}(\mathbf{I}) = (\pi, \nu')$), where σ' (resp., ν') is shown in Table 6 (resp., 7). □

Table 6: σ' resulting from projection

c	$\sigma'(c)$
Package	[Origin: string]
Letter	[Origin: string]
Box	[Origin: string, Contents: {string}]
Tube	[Origin: string, Contents: {string}]
Priority	[Origin: string, Time: [[time]]]
Express_saves	[Origin: string, Time: [[time]]]
One-transfer	[Origin: string, Contents: {string}, Time: [[time]]]
Two-transfer	[Origin: string, Contents: {string}, Time: [[time]]]

Table 7: ν' resulting from projection

o	$\nu'(o)$
o_3	[Origin: Rome, Time: $\{((8,00),[.45,.5]),((8,10),[.4,.5])\}$]
o_5	[Origin: Paris, Contents: {photos, books}, Time: $\{((12,00),[.3,.4]),((12,05),[.4,.7])\}$]

5.4 Extraction

The extraction operation eliminates some classes from a TPOB-schema, and also removes all objects in these classes from a TPOB-instance. This is often useful when we wish to focus interest on a certain part of the TPOB-schema (e.g. the part consisting of letters and its subclasses only) and/or TPOB-instances. We first define extraction on TPOB-schemas and then extend it to TPOB-instances.

5.4.1 Extraction on TPOB-Schemas

The extraction operator removes a set $\mathbf{C} \subseteq \mathcal{C}$ of classes from a TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \mathrm{me}, \wp)$. When defining extraction, care must be taken to ensure that the output schema does not introduce any class-subclass relationships that were not present in the original schema (nor should it delete any such relationships unless doing so is warranted by the deletion of classes in $\mathbf{C}$).

Formally, the *extraction* on a TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \mathrm{me}, \wp)$ with respect to a set of classes $\mathbf{C} \subseteq \mathcal{C}$, denoted $\Xi_{\mathbf{C}}(\mathbf{S})$, is the TPOB-schema $\mathbf{S}' = (\mathcal{C}', \sigma', \mathrm{me}', \wp')$, where:

- $\mathcal{C}' = \mathbf{C}$.
- σ' is the restriction of σ to $\mathcal{C}'$.

- For all $c \in \mathcal{C}'$, we define $\mathrm{me}'(c) = \{((X \cap \mathcal{C}') \cup X_{\mathcal{C}'}) \neq \emptyset \mid X \in \mathrm{me}(c)\}$, where $X_{\mathcal{C}'} = \{d_1 \in \mathcal{C}' \mid d_1 \Rightarrow d_2 \Rightarrow \cdots \Rightarrow d_k$ for some $d_2, \ldots, d_{k-1} \in \mathcal{C} - \mathcal{C}'$ and $d_k \in X - \mathcal{C}'\}$, and we assume that $|X_{\mathcal{C}'}| \leq 1$.
- For all $c_1, c_2 \in \mathcal{C}'$, we define $\wp'(c_1, c_2) = \prod_{i=1}^{k-1} \wp(d_i, d_{i+1})$, where the d_i's are such that $d_1 \Rightarrow d_2 \Rightarrow \cdots \Rightarrow d_k$, $d_1 = c_1$, $d_k = c_2$, and $d_2, \ldots, d_{k-1} \in \mathcal{C} - \mathcal{C}'$.

The following example illustrates the use of the extraction operator on TPOB-schemas.

Example 5.11 Consider the fully inherited TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \mathrm{me}, \wp)$ of Example 4.2. The extraction on $\mathbf{S}$ w.r.t. the set of classes $\mathbf{C} = \{\mathsf{Package}, \mathsf{Letter}, \mathsf{Priority}, \mathsf{One\text{-}transfer}\}$ is given by the TPOB-schema $\Xi_{\mathbf{C}}(\mathbf{S}) = (\mathcal{C}', \sigma', \mathrm{me}', \wp')$, where:

- $\mathcal{C}' = \{\mathsf{Package}, \mathsf{Letter}, \mathsf{Priority}, \mathsf{One\text{-}transfer}\}$.
- The type assignment σ' is given in Table 8.
- The probability assignment $\wp'$ is given in Figure 3.
- $\mathrm{me}'(\mathsf{Package}) = \{\{\mathsf{Letter}, \mathsf{One\text{-}transfer}\}, \{\mathsf{Priority}\}\}$, $\mathrm{me}'(\mathsf{Priority}) = \{\{\mathsf{One\text{-}transfer}\}\}$, and $\mathrm{me}'(\mathsf{Letter}) = \mathrm{me}'(\mathsf{One\text{-}transfer}) = \emptyset$. □

Table 8: Type assignment σ' resulting from extraction

c	$\sigma'(c)$
Package	[Origin : string, Destination : string, Delivery : [[time]]]
Letter	[Origin : string, Destination : string, Delivery : [[time]], Height : float, Width : float]
Priority	[Origin : string, Destination : string, Delivery : [[time]], Time : [[time]]]
One-transfer	[Origin : string, Destination : string, Delivery : [[time]], Height : float, Width : float, Depth : float, Contents : {string}, Time : [[time]], City : string, Arrive : [[time]], Shipment : [[time]]]

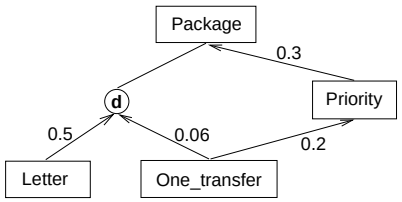

Figure 3: Class hierarchy and probability assignment resulting from extraction

5.4.2 Extraction on TPOB-instances

The extraction on a TPOB-instance $\mathbf{I}$ with respect to a set of classes $\mathbf{C}$ returns the TPOB-instance $\mathbf{I}'$ that contains all the objects (and their values) in $\mathbf{I}$ that belong to the classes in $\mathbf{C}$. Formally, the

extraction on a TPOB-instance $\mathbf{I} = (\pi, \nu)$ over a TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \mathrm{me}, \wp)$ with respect to a set of classes $\mathbf{C} \subseteq \mathcal{C}$, denoted $\Xi_{\mathbf{C}}(\mathbf{I})$, is the TPOB-instance (π', ν') over the TPOB-schema $\Xi_{\mathbf{C}}(\mathbf{S}) = (\mathcal{C}', \sigma', \mathrm{me}', \wp')$, wher π' is the restriction of π to $\mathcal{C}'$ and ν' is the restriction of ν to $\pi'(\mathcal{C}')$.

The following example illustrates the extraction on TPOB-instances.

Example 5.12 Consider the fully inherited TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \mathrm{me}, \wp)$ of Example 4.2 and the TPOB-instance $\mathbf{I} = (\pi, \nu)$ over $\mathbf{S}$ of Example 4.3. The extraction on $\mathbf{I}$ with respect to the set of classes $\mathbf{C} = \{\mathsf{PackageLetter}, \mathsf{Priority}, \mathsf{One\text{-}transfer}\}$ results in the TPOB-instance $\Xi_{\mathbf{C}}(\mathbf{I}) = (\pi', \nu')$ over the TPOB schema $\Xi_{\mathbf{C}}(\mathbf{S})$ in Example 5.11, where π' is given by Table 9 and ν' is given by $\nu'(o_3) = \nu(o_3)$. □

Table 9: π' and $(\pi')^*$ resulting from extraction

c	$\pi'(c)$	$(\pi')^*(c)$
Package	$\{\}$	$\{o_3\}$
Letter	$\{\}$	$\{\}$
Priority	$\{o_3\}$	$\{o_3\}$
One-transfer	$\{\}$	$\{\}$

6 TPOB-Algebra: Binary Operations

We now defi ne the natural join and Cartesian product operations. Our technical report [6] also contains defi nitions of condition join, intersection, union, and difference. Unless specifi ed otherwise, we assume all input TPOB-schemas to be fully inherited.

6.1 Natural Join

Our *natural join* operator (like its relational counterpart) forms a Cartesian product of two TPOB-instances, performs a selection forcing equality on attributes common to both TPOB-schemes, and then removes duplicate values. We fi rst defi ne natural join of two TPOB-schemas. We then defi ne the natural join of two values of probabilistic tuple types. Finally, we introduce the natural join of two TPOB-instances. In the rest of this section, let $\mathbf{S}_1 = (\mathcal{C}_1, \sigma_1, \mathrm{me}_1, \wp_1)$ and $\mathbf{S}_2 = (\mathcal{C}_2, \sigma_2, \mathrm{me}_2, \wp_2)$ be two TPOB-schemas.

6.1.1 Natural Join of TPOB-Schemas

Informally, the TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \mathrm{me}, \wp)$ produced by the natural join of the TPOB-schemas $\mathbf{S}_1$ and $\mathbf{S}_2$ is obtained as follows. First, the set of classes $\mathcal{C}$ is the Cartesian product of the sets of

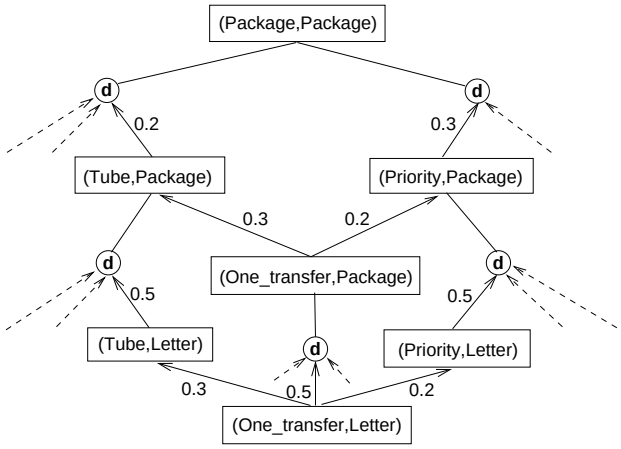

Figure 4: Natural join of schemas

classes $\mathcal{C}_1$ and $\mathcal{C}_2$. Second, the type $\sigma(c)$ of each class $c = (c_1, c_2) \in \mathcal{C}$ is given by the probabilistic tuple type containing every top-level attribute with its associated type in $\sigma_1(c_1)$ and $\sigma_2(c_2)$. Note that this assumes that every common top-level attribute of $\sigma_1(c_1)$ and $\sigma_2(c_2)$ has the same type in $\sigma_1(c_1)$ and $\sigma_2(c_2)$. We say $\mathbf{S}_1$ and $\mathbf{S}_2$ are *natural-join-compatible* when this condition is satisfied for any two classes $c_1 \in \mathcal{C}_1$ and $c_2 \in \mathcal{C}_2$. Third, the class hierarchy in $\mathbf{S}$ is defined as the Cartesian product of the class hierarchies in $\mathbf{S}_1$ and $\mathbf{S}_2$. That is, every class $c_1 \in \mathcal{C}_1$ (resp., $c_2 \in \mathcal{C}_2$) is combined with the immediate subclass and disjointness relationships expressed by every $\mathrm{me}_2(c_2)$ with $c_2 \in \mathcal{C}_2$ (resp., $\mathrm{me}_1(c_1)$ with $c_1 \in \mathcal{C}_1$); every class $c_1 \in \mathcal{C}_1$ (resp., $c_2 \in \mathcal{C}_2$) is combined with every conditional probability $\wp_2(c_2, d_2)$ in $\mathbf{S}_2$ (resp., $\wp_1(c_1, d_1)$ in $\mathbf{S}_1$).

Formally if two TPOB-schemas $\mathbf{S}_1$ and $\mathbf{S}_2$ are natural-join-compatible, then the *natural join* of $\mathbf{S}_1$ and $\mathbf{S}_2$, denoted $\mathbf{S}_1 \bowtie \mathbf{S}_2$, is defined as the TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \mathrm{me}, \wp)$, where

- $\mathcal{C} = \mathcal{C}_1 \times \mathcal{C}_2$.
- For each $c = (c_1, c_2) \in \mathcal{C}$, the probabilistic tuple type $\sigma(c) = [A_1 : \tau_1, \ldots, A_l : \tau_l]$ contains exactly every $A_i : \tau_i$ that belongs to either the type $\sigma_1(c_1)$ or the type $\sigma_2(c_2)$.
- For each $c = (c_1, c_2) \in \mathcal{C}$: $\mathrm{me}(c) = \{\{c_1\} \times \mathcal{P}_2 \mid \mathcal{P}_2 \in \mathrm{me}_2(c_2)\} \cup \{\mathcal{P}_1 \times \{c_2\} \mid \mathcal{P}_1 \in \mathrm{me}_1(c_1)\}$.
- For each immediate subclass relationship $(c_1, c_2) \Rightarrow (c_1, d_2)$ (resp., $(c_1, c_2) \Rightarrow (d_1, c_2)$) in $\mathbf{S}$, define $\wp((c_1, c_2), (c_1, d_2)) = \wp_2(c_2, d_2)$ (resp., $\wp((c_1, c_2), (d_1, c_2)) = \wp_1(c_1, d_1)$).

The following example illustrates the natural join of TPOB-schemas via the Package Example.

Example 6.1 Let $\mathbf{S}_1$ and $\mathbf{S}_2$ be the TPOB-schemas of Examples 4.2 and 5.11, respectively. Their natural join $\mathbf{S}_1 \bowtie \mathbf{S}_2$ is the TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \mathrm{me}, \wp)$ partially shown in Table 10 and Figure 4. □

Table 10: Type assignment σ resulting from natural join

c	$\sigma(c)$
(Package, Package)	[Origin: string, Destination: string, Delivery: [[time]]]
(Tube, Package)	[Origin: string, Destination: string, Delivery: [[time]], Height: float, Width: float, Depth: float, Contents: {string}]
(Priority, Package)	[Origin: string, Destination: string, Delivery: [[time]], Time: [[time]]]
(One-transfer, Letter)	[Origin: string, Destination: string, Delivery: [[time]], Height: float, Width: float, Depth: float, Contents: {string}, Time: [[time]], City: string, Arrive: [[time]], Shipment: [[time]]]

6.1.2 Intersection and Natural Join of Values

The natural join of two classical relations R and S contains one tuple for each pair of tuples $(r, s) \in R \times S$ that have the same values in the common attributes of R and S. Similarly, the natural join of two TPOB-instances $\mathbf{I}_1{=}(\pi_1, \nu_1)$ and $\mathbf{I}_2{=}(\pi_2, \nu_2)$ contains one object o for each pair of objects o_1 in $\mathbf{I}_1$ and o_2 in $\mathbf{I}_2$ such that the natural join of $\nu_1(o_1)$ and $\nu_2(o_2)$ (their concatenation combined with the intersection of the values in the common attributes) is defined, which then forms the value $\nu(o)$ of the new object o.

To define the natural join of two values v_1 and v_2 of probabilistic tuple types τ_1 and τ_2, respectively, we first define the *intersection* of two values v_1 and v_2 (of the same classical or probabilistic type τ) under the conjunction strategy $\otimes$, denoted $v_1 \cap_\otimes v_2$.

- If τ is a classical type and $v_1 = v_2$, then $v_1 \cap_\otimes v_2 = v_1$.
- If τ is an atomic probabilistic type and $w \neq \emptyset$, then $v_1 \cap_\otimes v_2 = w$, where $w = \{(v, I_1 \otimes I_2) \mid (v, I_1) \in v_1,\ (v, I_2) \in v_2\}$.
- If τ is a probabilistic tuple type over the set of top-level attributes $\mathbf{A}$ and all $v_1.A \cap_\otimes v_2.A$ are defined, then $(v_1 \cap_\otimes v_2).A = v_1.A \cap_\otimes v_2.A$ for all $A \in \mathbf{A}$.

Otherwise, $v_1 \cap_\otimes v_2$ is undefined. The following example illustrates the above concept.

Example 6.2 Let time be the standard calendar w.r.t. $hour \sqsupseteq minute$, and let $\otimes$ be a conjunction strategy. Consider the values $v_1{=}v_2{=}(12, 30)$ and $v_3{=}(12, 40)$ of the classical type time. Then, $v_1 \cap_\otimes v_2 = v_1$, while $v_1 \cap_\otimes v_3$ is undefined. Now consider the following values of the probabilistic type [[time]]:

$$v_1 = \{((9, 00), [.3, .5]), ((10, 00), [.3, .6]), ((11, 00), [.2, .5])\},\ v_2 = \{((12, 00), [.2, .4])\},$$
$$v_3 = \{((9, 00), [.3, .4]), ((11, 00), [.3, .6]), ((12, 00), [.2, .4])\}.$$

Then, $v_1 \cap_{\otimes_{in}} v_2$ is undefined, while $v_1 \cap_{\otimes_{in}} v_3 = \{((9, 00), [.09, .2]), ((11, 00), [.06, .3])\}$. □

We are now ready to define the natural join of two values v_1 and v_2 of probabilistic tuple types τ_1 and τ_2, respectively. Let $\otimes$ be a conjunction strategy. Let $\mathbf{A}_1$ and $\mathbf{A}_2$ be the top-level attributes of

τ_1 and τ_2, respectively, and let $\mathbf{A} = \mathbf{A}_1 \cap \mathbf{A}_2$. Let all $A \in \mathbf{A}$ have the same types in τ_1 and τ_2. The *natural join* of v_1 and v_2 under $\otimes$, denoted $v_1 \bowtie_\otimes v_2$, is defined as:

- $(v_1 \bowtie_\otimes v_2).A = v_i.A$ for all $A \in \mathbf{A}_i - \mathbf{A}$, $i \in \{1,2\}$, $(v_1 \bowtie_\otimes v_2).A = v_1.A \bowtie_\otimes v_2.A$ for all $A \in \mathbf{A}$. If all $v_1.A \bowtie_\otimes v_2.A$ with $A \in \mathbf{A}$ are defined, then $v_1 \bowtie_\otimes v_2$ is defined.

6.1.3 Natural Join of TPOB-Instances

Let $\mathbf{I}_1 = (\pi_1, \nu_1)$ and $\mathbf{I}_2 = (\pi_2, \nu_2)$ be two TPOB-instances over the natural-join-compatible TPOB-schemas $\mathbf{S}_1$ and $\mathbf{S}_2$, respectively. For $i \in \{1,2\}$, let $\mathbf{A}_i$ denote the set of top-level attributes of $\mathbf{S}_i$. Let $\otimes$ be a conjunction strategy. The *natural join* of $\mathbf{I}_1$ and $\mathbf{I}_2$ under $\otimes$, denoted $\mathbf{I}_1 \bowtie_\otimes \mathbf{I}_2$, is the TPOB-instance (π, ν) over $\mathbf{S}_1 \bowtie \mathbf{S}_2$, where:

- $\pi(c) = \{(o_1, o_2) \in \pi_1(c_1) \times \pi_2(c_2) \mid \nu_1(o_1) \bowtie_\otimes \nu_2(o_2) \text{ is defined}\}$, for all $c = (c_1, c_2) \in \mathcal{C}_1 \times \mathcal{C}_2$.
- $\nu(o) = \nu_1(o_1) \bowtie_\otimes \nu_2(o_2)$, for all $o = (o_1, o_2) \in \pi(\mathcal{C}_1 \times \mathcal{C}_2)$.

Example 6.3 Let $\mathbf{S}_1$ and $\mathbf{S}_2$ be the TPOB-schemas given in Example 4.2 and created in Example 5.11, respectively. Let $\mathbf{I}_1$ and $\mathbf{I}_2$ be the TPOB-instances over $\mathbf{S}_1$ and $\mathbf{S}_2$ produced in Examples 5.9 and 5.12, respectively. The natural join of $\mathbf{I}_1$ and $\mathbf{I}_2$ under the ignorance conjunction strategy $\otimes_{ig}$ is the TPOB-instance $\mathbf{I}_1 \bowtie_{\otimes_{ig}} \mathbf{I}_2 = (\pi, \nu)$ over $\mathbf{S}_1 \bowtie \mathbf{S}_2 = (\mathcal{C}, \sigma, \mathrm{me}, \wp)$, where π is given by $\pi((\mathsf{Priority}, \mathsf{Priority})) = \{(o_3, o_3)\}$ and $\pi(c) = \emptyset$ for all other classes $c \in \mathcal{C}$, and ν is given by Table 11. □

Table 11: ν resulting from natural join

o	$\nu(o)$
(o_3, o_3)	[Origin: Rome, Destination: Boston, Delivery: $\{((18,00),[.4,1]),((18,31),[.3,1])\}$, Time: $\{((8,00),[.45,1]),((8,10),[.4,1])\}$]

6.2 Cartesian Product

In the above definition of natural join, if the sets $\mathbf{A_1}$ and $\mathbf{A_2}$ are disjoint, then the natural join is called *Cartesian product* and denoted by the symbol $\times$. Two TPOB-schemas $\mathbf{S}_1$ and $\mathbf{S}_2$ are *Cartesian-product-compatible* iff they can be combined using Cartesian product, that is, iff for all classes c_1 in $\mathbf{S}_1$ and c_2 in $\mathbf{S}_2$, the types of c_1 and c_2 have disjoint sets of top-level attributes.

7 Preservation of Consistency and Coherence

We now show that all algebraic operators defined in Sections 5 and 6 preserve consistency and coherence of schemas and instances. If the input TPOB-schemas (resp., TPOB-instances) are consistent (resp., coherent), then the output TPOB-schemas (resp., TPOB-instances) are also consistent (resp., coherent).

The operators of selection and restricted selection trivially preserve consistency of schemas, as the input schemas coincide with the output schemas. Projection also preserves consistency of schemas, as it only modifies type assignments. The following result shows that extraction and natural join, and thus also Cartesian product, preserve consistency of schemas.

Theorem 7.1 *Let* $\mathbf{S}$ *be a TPOB-schema, and let* $\mathbf{C}$ *be a set of classes from* $\mathbf{S}$*. Let* $\mathbf{S}_1$ *and* $\mathbf{S}_2$ *be two natural-join-compatible TPOB-schemas.*

(a) If $\mathbf{S}$ *is consistent, then* $\Xi_{\mathbf{C}}(\mathbf{S})$ *is consistent.*

(b) If $\mathbf{S}_1$ *and* $\mathbf{S}_2$ *are consistent, then* $\mathbf{S}_1 \bowtie \mathbf{S}_2$ *is consistent.*

The operations of selection and restricted selection preserve coherence of instances, as they do not modify the input TPOB-schemas and they may only modify the input TPOB-instances by removing objects and changing value assignments to objects. Similarly, projection preserves coherence of instances, as it may only modify type and value assignments to classes and objects, respectively. The result below shows that natural join and extraction operation preserves coherence of instances when we do not remove any characteristic classes.

Theorem 7.2 *Let* $\mathbf{I}$*,* $\mathbf{I}_1$*, and* $\mathbf{I}_2$ *be TPOB-instances over the TPOB-schemas* $\mathbf{S}$*,* $\mathbf{S}_1$*, and* $\mathbf{S}_2$*, respectively, where* $\mathbf{I} = (\pi, \nu)$ *and* $\mathbf{S} = (\mathcal{C}, \sigma, \mathrm{me}, \wp)$*. Let* $\mathbf{C} \subseteq \mathcal{C}$ *such that* $\{c \in \mathcal{C} \mid c$ *is characteristic for* $\mathrm{ext}(d)(o)$ *for some* $d \in \mathbf{C}$ *and some* $o \in \pi(\mathbf{C})\} \subseteq \mathbf{C}$*. Let* $\mathbf{S}_1$ *and* $\mathbf{S}_2$ *be natural-join-compatible.*

(a) If $\mathbf{I}$ *is coherent, then* $\Xi_{\mathbf{C}}(\mathbf{I})$ *is coherent.*

(b) If $\mathbf{I}_1$ *and* $\mathbf{I}_2$ *are coherent, then* $\mathbf{I}_1 \bowtie \mathbf{I}_2$ *is coherent.*

8 Implicit TPOB-Instances

We now focus on how to efficiently implement the TPOB-instances introduced in Section 4 and the algebraic operations on TPOB-instances presented in Sections 5 and 6.

The main problem with TPOB-instances is that the size of a TPOB-instance can be very large. Table 4 shows that a probability must be associated with each time point involved. But to state that a given package will arrive in St. Louis sometime between 5:30 pm and 6:30 pm (if we reason at

a minute by minute level) requires 60 time points to be specified (Table 4 only shows a couple of time points). The number of time points increase even more if they are defined in a calendar based on seconds instead of minutes. The second problem is that the large size of the TPOB-instances causes the cost of the algebraic operations to be potentially high.

In this and the following section, we alleviate these problems by defining *implicit* TPOB-instances, together with algebraic operations on implicit TPOB-instances. An implicit TPOB-instance succinctly captures a potentially large TPOB value. We further show that these implicit operations correctly implement their counterparts in Sections 5 and 6 by *solely considering the implicit (i.e., succinct) TPOB-instance rather than the explicit one*. Hence, all our implicit algebraic operators are usually much more efficient than their explicit counterparts. Furthermore, they immediately preserve consistency and coherence of schemas and instances, respectively.

The main idea behind implicit TPOB-instances is to use an implicit and thus more compact representation of the values of probabilistic types. We generalize a constraint-based approach due to Dekhtyar et al. [13].

For example, consider an explicit value of the form $v = \{(1, [0.001, 0.001]), \ldots, (1000, [0.001, 0.001)\}$. This says that v's value is one of $1, \ldots, 1000$ with some associated probability intervals. Instead of storing 1000 pairs of the form $(i, [0.001, 0.001])$, we could describe the range of values via the constraint $1 \leq t \leq 1000$. We can store the probability information in this case by merely saying that the uniform distribution is used to distribute probability values over the values $\{1, \ldots, 1000\}$ and that the entire probability mass 1 is distributed. Thus, our general idea is to use a constraint C to represent the time points at which some event may possibly hold, a lower and upper probability bound l, u, and a distribution δ. The technical definition of an *implicit value* will use these intuitions, but will include some additional twists.

We now first describe the concepts of constraints and of probability distribution functions. We then define implicit values of probabilistic types and implicit TPOB-instances.

Constraints. We use a constraint to implicitly specify a set of valid time points of a calendar, or a set of values of a classical type with a totally ordered domain, which is given by the set of all solutions to that constraint.

An *atomic constraint* for a calendar τ over the linear temporal hierarchy $T_1 \sqsupseteq \cdots \sqsupseteq T_n$ is either an *atomic time-value constraint* ($T_i\, \theta\, v_i$), where $\theta \in \{\leq, <, =, \neq, >, \geq\}$ and v_i is a time value of T_i, or an *atomic time-interval constraint* ($t_1 \sim t_2$), where $t_1, t_2 \in \mathrm{dom}(\tau)$ and $t_1 \leq_H t_2$. An *atomic constraint* for a classical type τ with a linear order $<$ on the domain $\mathrm{dom}(\tau)$ is either of the form $(\theta\, v)$, where $\theta \in \{\leq, <, =, \neq, >, \geq\}$ and $v \in \mathrm{dom}(\tau)$, or of the form $(v_1 \sim v_2)$, where $v_1, v_2 \in \mathrm{dom}(\tau)$ with $v_1 \leq v_2$. We use (t_1) (resp., (v_1)) to abbreviate $(t_1 \sim t_1)$ (resp., $(v_1 \sim v_1)$). A

constraint for the above forms of classical types τ is a Boolean combination of atomic constraints for τ (that is, constructed from atomic constraints for τ by using the Boolean operators $\neg$ and $\wedge$).

We now define the semantics of a constraint, that is, the set of time points (resp., values) that it specifies. A time point $s = (s_1, \ldots, s_n) \in \mathrm{dom}(\tau)$ is a *solution* to an atomic constraint $(T_i\, \theta\, v_i)$ (resp., $(t_1 \sim t_2)$) for τ, denoted $s \models (T_i\, \theta\, v_i)$ (resp., $s \models (t_1 \sim t_2)$), iff $s_i\, \theta\, v_i$ (resp., $t_1 \leq_H s \leq_H t_2$). A value $s \in \mathrm{dom}(\tau)$ is a *solution* to an atomic constraint $(\theta\, v)$ (resp., $(v_1 \sim v_2)$) for τ, denoted $s \models (\theta\, v)$ (resp., $s \models (v_1 \sim v_2)$), iff $s\, \theta\, v$ (resp., $v_1 \leq s \leq v_2$). The solutions $s \in \mathrm{dom}(\tau)$ to a constraint C for τ are inductively defined by (i) $s \models \neg C_1$ iff it is not the case that $s \models C_1$, and (ii) $s \models (C_1 \wedge C_2)$ iff $s \models C_1$ and $s \models C_2$, for all constraints C_1, C_2. We use $\mathrm{sol}(C)$ to denote the set of all solutions $s \in \mathrm{dom}(\tau)$ to the constraint C for τ.

Probability Distribution Functions. Suppose S is a nonempty set of pairwise mutually exclusive events $s_1, \ldots, s_n$. A *probability distribution function (PDF)* (or *distribution function*) over S is a mapping $\rho\colon S \to [0, 1]$ such that $\sum_{s \in S} \rho(s) \leq 1$. The most widely used PDF is the *uniform distribution* over S, denoted U_S, which is the function $U_S\colon S \to [0, 1]$ defined by $U(s) = 1 \,/\, n$ for all $s \in S$. Our framework can work with any PDF whatsoever (more details are in [6]).

Implicit Values of Probabilistic Types. An implicit value of a probabilistic type is a finite set of implicit tuples [13]. An example of an implicit tuple (which we define formally below) is $(1 \leq t \leq 40, 1 \leq t \leq 100, 0.5, 1, u)$. This implicit tuple says that the valid time points are solutions of the first constraint $1 \leq t \leq 40$ and that a probability mass of 0.5 to 1 is uniformly distributed over the solutions of the second constraint $1 \leq t \leq 100$. From this, we can infer that there is a probability mass of 0.2 to 0.4 over the set of valid time points $\{1, \ldots, 40\}$. Using this intuition, we can now define implicit tuples formally.

Let τ be either a calendar or a classical type with a totally ordered domain. An *implicit tuple* for τ is a 5-tuple (C, D, l, u, δ), where C, D are constraints for τ with $\emptyset \subset \mathrm{sol}(C) \subseteq \mathrm{sol}(D)$, l, u are reals with $0 \leq l \leq u \leq 1$, and δ is a distribution function over $\mathrm{sol}(D)$. If $\mathrm{sol}(C) = \mathrm{sol}(D)$, then we use $(\#)$ to abbreviate C.

The constraint C describes the valid values, while D describes the values from which the probabilities for solutions of C are *derived*. When selection operations are performed, we can merely restrict C to the selection condition instead of computing a new distribution (which would be very expensive). Thus, the use of D eliminates the potentially expensive operation of computing new distributions every time a query is asked.

We now define *implicit values* of probabilistic types by induction as follows:

- An implicit value of an atomic probabilistic type $[\![\tau]\!]$ is a finite set of implicit tuples for τ.
- An implicit value of a probabilistic type $[A_1\colon \tau_1, \ldots, A_k\colon \tau_k]$ is of the form $[A_1\colon v_1, \ldots,$

$A_k : v_k]$, where $v_1, \ldots, v_k$ are either values or implicit values of $\tau_1, \ldots, \tau_k$.

In the sequel, the values of probabilistic types introduced in Section 3 are called *explicit values*, to better distinguish them from the above concept of implicit values of probabilistic types. An implicit value of a probabilistic type in the Package Example is given below.

Example 8.1 An implicit value of the atomic probabilistic type $[\![\mathsf{time}]\!]$, where time is the calendar over $hour \sqsupseteq minute$, is given by $v = ((\#), ((12,00) \sim (12,59)), 0.5, 1, U)$. □

Observe now that every implicit value v of an atomic probabilistic type $[\![\tau]\!]$ has a unique equivalent explicit value $\varepsilon(v)$, which is defined by $\varepsilon(v) = \{(v', [l \cdot \delta(v'), u \cdot \delta(v')]) \mid \exists (C, D, l, u, \delta) \in v : v' \in \mathrm{sol}(C)\}$. ε is an *implicit to explicit transformation*. Conversely, every explicit value of $[\![\tau]\!]$ has at least one equivalent implicit value. The following example illustrates the relationship between implicit values v and their equivalent explicit values $\varepsilon(v)$.

Example 8.2 The implicit value v given in Example 8.1 has the equivalent explicit value $\varepsilon(v) = \{((12,00), [\frac{0.5}{60}, \frac{1}{60}]), ((12,01), [\frac{1}{60}]), ((12,02), [\frac{1}{60}]), \ldots, ((12,59), [\frac{1}{60}])\}$. *Note that instead of explicitly expressing a probability interval for each of 60 time points, the implicit value shown in Example 8.1 captures this explicit information via a single statement.* Hence, implicit values are far more succinct than explicit ones. □

As every implicit value has its explicit counterpart, we can easily extend the notion of consistency to implicit values as follows. An implicit value $v = \{(C_1, D_1, l_1, u_1, \delta_1), \ldots, (C_n, D_n, l_n, u_n, \delta_n)\}$ of an atomic probabilistic type is *consistent* iff $\mathrm{sol}(C_i \wedge C_j) = \emptyset$ for all $i, j \in \{1, \ldots, n\}$ with $i \neq j$, and it holds that $\sum_{i=1}^{n} \sum_{v_i \in \mathrm{sol}(C_i)} l_i \cdot \delta_i(v_i) \leq 1 \leq \sum_{i=1}^{n} \sum_{v_i \in \mathrm{sol}(C_i)} u_i \cdot \delta_i(v_i)$. An implicit value v of a probabilistic type is *consistent* iff all contained implicit values of atomic probabilistic types are consistent.

8.1 Implicit TPOB-Instances

Implicit TPOB-instances associate an implicit value with each object (rather than an explicit one). Formally, an *implicit TPOB-instance* over a consistent TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \mathrm{me}, \wp)$ is a pair $\mathbf{I} = (\pi, \nu)$, where (i) $\pi : \mathcal{C} \to 2^{\mathcal{O}}$ maps each class c to a finite subset of $\mathcal{O}$ such that $\pi(c_1) \cap \pi(c_2) = \emptyset$ for all distinct $c_1, c_2 \in \mathcal{C}$, and (ii) ν maps each oid $o \in \pi(c)$, $c \in \mathcal{C}$, to an implicit value of type $\sigma^\star(c)$. We then define $\pi(\mathcal{C})$, the mapping $\pi^\star$, the concept of a probabilistic extent of a class, and the notions of coherence and consistency for implicit TPOB-instances as in Section 4.3. In the sequel, we write *explicit TPOB-instance* to refer to the TPOB-instances defined in Section 4.3.

Every implicit TPOB-instance $\mathbf{I}$ has a unique equivalent explicit TPOB-instance $\varepsilon(\mathbf{I})$, which is obtained from $\mathbf{I}$ by replacing each contained implicit value v of an atomic probabilistic type by

its explicit counterpart $\varepsilon(v)$. Conversely, every explicit TPOB-instance has at least one equivalent implicit TPOB-instance.

Example 8.3 An implicit TPOB-instance $\mathbf{I} = (\pi, \nu)$ over the TPOB-schema $\mathbf{S}$ of Example 4.1 is given by the mapping π in Table 3 and the value assignment ν in Table 12. □

Table 12: Value assignment ν

o	$\nu(o)$
o_3	[Origin: Rome, Destination: Boston, Delivery: $\{((\#), ((18,00) \sim (18,31)), .4, .9, U)$, $((\#), ((12,00) \sim (12,31)), .2, .3, G)\}$, Time: $\{((\#), ((8,00) \sim (8,10)), .4, 1, G)\}$]
o_5	[Origin: Paris, Destination: San_Jose, Delivery: $\{((\#), ((12,00) \sim (12,15)), .15, 1, U)\}$, Height: 60, Width: 50, Depth: 40, Contents: {photos, books}, Time: $\{(((12,00) \sim (12,05)), .3, 1, G)\}$, STOPone: [City: New_York, Arrive: $\{(((14,00) \sim (14,05)) \vee ((14,25) \sim (14,30)), ((14,00) \sim (14,30)), .3, 1, U)\}$, Shipment: $\{((\#), ((16,00) \sim (17,00)), .6, 1, U)\}$], STOPtwo: [City: ST_Louis, Arrive: $\{((\#), ((17,30) \sim (17,45)), .5, 1, U)\}$, Shipment: $\{((\#), ((18,00) \sim (18,30)), .6, 1, U)\}$]]

9 The Implicit Algebra

In this section, we define selection, restricted selection, and natural join. Projection, extraction, Cartesian product, and condition join are independent of the kind of values of each object, and thus defined in exactly the same way for explicit and implicit TPOB-instances. We show that each operation of the implicit TPOB-algebra correctly implements its explicit counterpart. For unary algebraic operators *op*, the correctness theorems are of the form $\varepsilon(op^i(\mathbf{I})) = op^e(\varepsilon(\mathbf{I}))$ — cf. Fig. 1. op^i (resp. op^e) denotes the implicit (resp. explicit) versions of the operator *op*. Similar correctness results are also shown to hold for binary algebraic operators. Whenever we apply an algebraic operation *op*, we can tell by the type of argument (explicit TPOB instance or implicit) whether the operator is one from the explicit or from the implicit algebra. Hence, we will usually just write *op* instead of op^e, op^i as this can be determined from context. Other operators are contained in [6].

9.1 Selection

In order to define the selection operation on implicit TPOB-instances, it is sufficient to define how to evaluate path expressions and how to assess the probability that an implicit value satisfies an atomic selection condition. The valuation of selection conditions, the satisfaction of probabilistic selection conditions, and the selection on implicit TPOB-instances are then defined in the same way as in Section 5.1.2.

We first define how to evaluate path expressions. Let P be a path expression for the probabilistic type τ. The *valuation* of P under an implicit value v of τ, denoted $v.P$, is defined by:

- if $v = [A_1 : v_1, \ldots, A_k : v_k]$ and $P = A_i$, then $v.P = v_i$;
- if $v = [A_1 : v_1, \ldots, A_k : v_k]$ and $P = A_i.R$, then $v.P = v_i.R$;
- If $v = \{(C_1, D_1, l_1, u_1, \delta_1), \ldots, (C_k, D_k, l_k, u_k, \delta_k)\}$ and $P = [\![R]\!]$, then $v.P = \{(C_1, D_1, l_1, u_1, \delta_1, R), \ldots, (C_k, D_k, l_k, u_k, \delta_k, R)\}$. We call such sets *generalized implicit values* of τ ;
- Otherwise, $v.P$ is undefined.

Given an implicit TPOB-instance $\mathbf{I} = (\pi, \nu)$ over a TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \mathrm{me}, \wp)$, we now define the meaning of an ASC ϕ w.r.t. an object o in $\mathbf{I}$. As in Section 5.1.2, we associate with each ϕ a closed subinterval of $[0, 1]$, which describes the range for the probability that the object o in $\mathbf{I}$ satisfies ϕ. This probability is computed in a similar way as in Section 5.1.2, based on the translation ε from implicit to explicit values. More formally, the *probabilistic valuation* of ϕ w.r.t. $\mathbf{I}$ and $o \in \pi(\mathcal{C})$, denoted $\mathrm{prob}_{\mathbf{I},o}$, is defined as follows (where $\oplus$ is the disjunction strategy for mutual exclusion):

- $\mathrm{prob}_{\mathbf{I},o}(in(c)) = [\min(\mathrm{ext(c)}(o)), \max(\mathrm{ext(c)}(o))]$.
- Let P be a path expression for the type of o. If $\nu(o).P$ is a value of a classical type, then define $V = \{((\#), (\nu(o)), 1, 1, U, P)\}$, else if $\nu(o).P$ is a generalized implicit value of an atomic probabilistic type, then define $V = \nu(o).P$. Otherwise, V is undefined.

$$\mathrm{prob}_{\mathbf{I},o}(P\,\theta\,v) = \begin{cases} \bigoplus_{i=1}^{k} I_i & \text{if } V \text{ is defined} \\ \text{undefined} & \text{otherwise,} \end{cases}$$

  where $I_1, \ldots, I_k$ are the intervals $[l \cdot \delta(w), u \cdot \delta(w)]$ such that $(C, D, l, u, \delta, S) \in V$, $w \in \mathrm{sol}(C)$, and $w.S\,\theta\,v$, if V is defined. Note that $\mathrm{prob}_{\mathbf{I},o}(P\,\theta\,v)$ is undefined, if some $w.S\,\theta\,v$ is undefined.

- For each $i \in \{1, 2\}$, let P_i be a path expression for the type of o. If $\nu(o).P_i$ is a value of a classical type, then define $V_i = \{((\#), (\nu(o)), 1, 1, U, P_i)\}$, else if $\nu(o).P_i$ is a generalized implicit value of an atomic probabilistic type, then define $V_i = \nu(o).P_i$. Otherwise, V_i is undefined. Then,

$$\mathrm{prob}_{\mathbf{I},o}(P_1\,\theta_{\otimes}\,P_2) = \begin{cases} \bigoplus_{i=1}^{k} I_i & \text{if } V_1 \text{ and } V_2 \text{ are defined} \\ \text{undefined} & \text{otherwise,} \end{cases}$$

  where $I_1, \ldots, I_k$ is the list of all intervals $[l_1 \cdot \delta_1(v_1), u_1 \cdot \delta_1(v_1)] \otimes [l_2 \cdot \delta_2(v_2), u_2 \cdot \delta_2(v_2)]$ such that $(C_i, D_i, l_i, u_i, \delta_i, S_i) \in V_i$, $v_i \in \mathrm{sol}(C_i)$, and $v_1.S_1\,\theta\,v_2.S_2$, if V_1 and V_2 are defined. Observe that $\mathrm{prob}_{\mathbf{I},o}(P_1\,\theta_{\otimes}\,P_2)$ is undefined, if some $v_1.S_1\,\theta\,v_2.S_2$ is undefined.

The following example shows some probabilistic valuations of atomic selection conditions.

Example 9.1 Consider the implicit TPOB-instance $\mathbf{I} = (\pi, \nu)$ of Example 8.3. The ASC $\phi = \mathsf{Delivery} > (18, 25)$ is assigned $[0, 0]$ and $[.075, .169]$ under $\mathrm{prob}_{\mathbf{I},o_5}$ and $\mathrm{prob}_{\mathbf{I},o_3}$, respectively. Here, $[.075, .169] = \bigoplus_{i=1}^{6}[.0125, .0281]$, where the six intervals $[.0125, .0281]$ are associated with the six time points $(18, 26), \ldots, (18, 31)$ of the value of object o_3 at the attribute Delivery. □

The following result shows that selection on implicit TPOB-instances correctly implements its counterpart on explicit TPOB-instances. That is, the mapping ε commutes with σ_ϕ.

Theorem 9.2 (correctness of selection) *Let* $\mathbf{I}$ *be an implicit TPOB-instance over a TPOB-schema* $\mathbf{S}$*, and let* ϕ *be a probabilistic selection condition. Then,* $\sigma_\phi(\varepsilon(\mathbf{I})) = \varepsilon(\sigma_\phi(\mathbf{I}))$.

9.2 Restricted Selection

In order to define the restricted selection operation on implicit TPOB-instances, it suffices to define the restricted selection on implicit values. Restricted selection on implicit TPOB-instances is then defined in the same way as in Section 5.2.

Let τ be a probabilistic tuple type, and let $v = [A_1\colon v_1, \ldots, A_i\colon v_i, \ldots, A_k\colon v_k]$ be an implicit value of τ. Let ϕ be an atomic selection condition of the form $P\,\theta\,w$, where P is a path expression for τ. The *restricted selection* on v with respect to ϕ, denoted $\sigma^r_\phi(v)$, is defined by:

- If v_i is a value of a classical type, $P = A_i$, and $v_i\,\theta\,w$, then $\sigma^r_\phi(v) = v$.
- If v_i is an implicit value of an atomic probabilistic type, and $P = A_i$, then $\sigma^r_\phi(v)$ is obtained from v by replacing v_i by $\{((\theta\,w) \wedge C, D, l, u, \delta) \mid (C, D, l, u, \delta) \in v_i,\ \mathrm{sol}((\theta\,w) \wedge C) \neq \emptyset\}$.
- If v_i is an implicit value of a probabilistic tuple type, and $P = A_i.R$, then $\sigma^r_\phi(v)$ is obtained from v by replacing v_i by $\sigma^r_{R\,\theta\,w}(v_i)$.
- Otherwise, $\sigma^r_\phi(v)$ is undefined.

Example 9.3 Consider the implicit TPOB-instance $\mathbf{I} = (\pi, \nu)$ of Example 8.3 and the atomic selection condition $\phi = \mathsf{Delivery} > (18, 25)$. The restricted selection on $\mathbf{I}$ with respect to ϕ is given by the implicit TPOB-instance $\sigma^r_\phi(\mathbf{I}) = (\pi', \nu')$, where $\pi'(\mathsf{Priority}) = \{o_3\}$, $\pi'(c) = \emptyset$ for all other classes c, and $\nu'(o_3) = [\mathsf{Origin}\colon\ \mathsf{Rome}, \mathsf{Destination}\colon \mathsf{Boston}, \mathsf{Delivery}\colon \{(((18, 26) \sim (18, 31)), ((18, 00) \sim (18, 31)), .4, .9, U)\}, \mathsf{Time}\colon \{((\#), ((8, 00) \sim (8, 10)), .4, 1, G)\}]$. □

The following theorem shows that restricted selection on implicit instances correctly implements its counterpart on explicit instances. That is, the mapping ε commutes with σ^r_ϕ.

Theorem 9.4 (correctness of restricted selection) *Let* $\mathbf{I}$ *be an implicit TPOB-instance over a TPOB schema* $\mathbf{S} = (\mathcal{C}, \sigma, \mathrm{me}, \wp)$*. Let* ϕ *be an atomic selection condition of the form* $P\,\theta\,v$*, where* P *is a path expression for every* $\sigma(c)$ *with* $c \in \mathcal{C}$*. Then,* $\sigma^r_\phi(\varepsilon(\mathbf{I})) = \varepsilon(\sigma^r_\phi(\mathbf{I}))$.

9.3 Natural Join

In order to define the natural join operation on implicit TPOB-instances, we need to define the intersection of two implicit values. The join of two implicit values and the join of two TPOB-instances are then defined in the same way as in Section 6.1.

Let v_1 and v_2 be either two values of the same classical type τ or two implicit values of the same probabilistic type τ, and let $\otimes$ be a conjunction strategy. The *intersection* of v_1 and v_2 under $\otimes$, denoted $v_1 \cap_\otimes v_2$, is inductively defined as follows:

- If τ is a classical type and $v_1 = v_2$, then $v_1 \cap_\otimes v_2 = v_1$.
- If τ is an atomic probabilistic type and $w \neq \emptyset$, then $v_1 \cap_\otimes v_2 = w$, where:

$$\begin{aligned} w \;=\; \{((\#), (v), l, u, U) \mid \exists (C_1, D_1, l_1, u_1, \delta_1) \in v_1, (C_2, D_2, l_2, u_2, \delta_2) \in v_2 : \\ v \in \mathrm{sol}(C_1 \wedge C_2),\ [l, u] = [l_1 \cdot \delta_1(v), u_1 \cdot \delta_1(v)] \otimes [l_2 \cdot \delta_2(v), u_2 \cdot \delta_2(v)]\} \,. \end{aligned}$$

- If τ is a probabilistic tuple type over the set of top-level attributes $\mathbf{A}$, and all $v_1.A \cap_\otimes v_2.A$ are defined, then $(v_1 \cap_\otimes v_2).A = v_1.A \cap_\otimes v_2.A$ for all $A \in \mathbf{A}$.
- Otherwise, $v_1 \cap_\otimes v_2$ is undefined.

The following example illustrates the intersection of two implicit values of atomic probabilistic types.

Example 9.5 Let time be the standard calendar w.r.t. $hour \sqsupseteq minute$, and let $\otimes_{in}$ be the conjunction strategy for independence. Consider now the following implicit values of the atomic probabilistic type $[\![\mathsf{time}]\!]$:

$$v_1 = \{(((9,41) \sim (9,45)), ((9,41) \sim (9,50)), .5, 1, U)\},\ v_2 = \{(\#), ((9,44) \sim (9,48)), .2, 1, U)\}$$

Then, $v_1 \cap_{\otimes_{in}} v_2 = \{((\#), ((9,44)), .04, .004, U), ((\#), ((9,45)), .04, .004, U)\}$. □

The result below shows that join on implicit TPOB-instances is correct, i.e. the mapping ε commutes with $\bowtie_\otimes$.

Theorem 9.6 (correctness of natural join) *Let* $\mathbf{I}_1$ *and* $\mathbf{I}_2$ *be two implicit TPOB-instances over the natural-join-compatible TPOB-schemas* $\mathbf{S}_1$ *and* $\mathbf{S}_2$*, respectively. Let* $\otimes$ *be a conjunction strategy. Then,* $\varepsilon(\mathbf{I}_1) \bowtie_\otimes \varepsilon(\mathbf{I}_2) = \varepsilon(\mathbf{I}_1 \bowtie_\otimes \mathbf{I}_2)$.

10 Conclusions

There are numerous applications where object models are naturally used and where temporal uncertainty is a critical part of the application. For example, shipping and transportation companies utilize airplanes, ships, and trucks, all with different capacities, fuel requirements, fuel consumption, servicing needs, and other properties and ship millions of items daily. The properties of these vehicles vary dramatically and the items being shipped also have diverse properties (commercial shippers such as CSX ship everything from packages of papers to machine parts, vehicles, and hazardous materials, each with very different properties). Such shippers use prediction programs that provide temporal projections that are characterized by uncertainty. These projections (and the other data described above) are useful in a wide range of decision making activities ranging from scheduling to asset allocation problems. The ability to query the above data in the presence of such temporally uncertain projections is critical for such decision making.

In this paper, we have made a first attempt to deal with temporal uncertainty in object-based systems. We have provided a data model and algebraic operations for such systems. Our framework allows us to specify the uncertainty about when certain events may occur, as well as uncertainty about what the object is. We have presented explicit TPOB-instances, where the sets of time points along with their probability intervals are enumerated, and implicit ones, where the sets of time points are expressed by constraints and their probability intervals by probability distribution functions. Thus, implicit object base instances are succinct representations of explicit ones; they allow for an efficient implementation of algebraic operations, while their explicit counterparts make defining algebraic operations easy and are easily understood by end users. As a consequence, we suggest that the implicit representation be used to implement TPOBs, but the answers to queries can be *presented* to users by converting the answer (in the implicit algebra) to its explicit counterpart. Finding a correct probability function can be a challenge for both the implicit and the explicit representation - if one has the probabilities for one representation, then the probabilities for the other can be easily inferred. We have defined our algebraic operations on both explicit and implicit object base instances, and shown that each operation on implicit object base instances correctly implements its counterpart on explicit instances.

There are numerous directions for future research. Building physical cost models and cost based query optimizers for TPOBs is a major challenge that must be addressed if applications such as the package and stock market example are to scale up for heavy duty use. Building mechanisms to update such databases poses yet another challenge. Building view creation and maintenance algorithms provides a third challenge. Developing an implementation of (the implicit version of) TPOBs poses a fourth major challenge as it will provide a testbed for all the algorithms resulting

from the other problems mentioned here.

Acknowledgements. We thank the reviewers for their excellent comments that greatly improved the paper. This work has been funded by ARL contract DAAL0197K0135, ARO grant DAAD1900-10484, DARPA/RL contract F306029910552, the ARL ADA CTA, by NSF grant 937756, by the Austrian Science Fund under project Z29-INF, by a DFG grant, and by a Marie Curie Individual Fellowship of the European Community programme "Human Potential" under contract number HPMF-CT-2001-001286 (Disclaimer: The authors are solely responsible for information communicated and the European Commission is not responsible for any views or results expressed).

Table of Commonly Used Symbols

Table 13: Notation

Symbol	Description	Section where Defined
$[\![\tau]\!]$	An atomic probabilistic type	3
$\mathcal{C}$	Set of classes	4.1
$\sigma(c)$	Type assignment of a class c	4.1
$\Rightarrow$	Subclass relationship	4.1
$\Rightarrow^\star$	The reflexive and transitive closure of $\Rightarrow$	4.1
$\mathrm{me}(c)$	Partition of the subclasses of a class c	4.1
$\wp(c_1, c_2)$	The conditional probability that an object belongs to a class c_1 given that it belongs to an immediate subclass c_2.	4.1
$\mathbf{S} = (\mathcal{C}, \sigma, \mathrm{me}, \wp)$	A TPOB-schema	4.1
ζ	A mapping from the set of classes to the sets of objects	4.1
$\pi(c)$	Set of object identifiers in a class c	4.3
$\nu(o)$	The value of an object $o \in \pi(c)$	4.3
$\mathbf{I} = (\pi, \nu)$	TPOB-instance	4.3
$\mathrm{ext(c)}(o)$	The probabilities with which an object o belongs to a class c	4.3
$\pi^\star(c)$	Object identifiers that are created and inherited in c	4.3
$\sigma^\star(c)$	Inheritance completion of a type of a class c	4.2
ε	Mapping from implicit values (instances) to explicit values (instances)	8.3

References

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

[2] S. Adali and L. Pigaty. The DARPA advanced logistics program. *Annals of Mathematics and Artificial Intelligence*, 37(4):409–452, 2003.

[3] E. I. Altman. Financial ratios, discriminant analysis, and the prediction of a corporate bankruptcy. *Journal of Finance*, 23:589–609, 1968.

[4] M. Atkinson, D. DeWitt, D. Maier, F. Bancilhon, K. Dittrich, and S. Zdonik. The object-oriented database system manifesto. In *Proceedings DOOD-89*, pp. 223–240. Elsevier Science Publishers, 1990.

[5] 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.

[6] V. Biazzo, R. Giugno, T. Lukasiewicz, and V. S. Subrahmanian. Temporal probabilistic object bases. Technical Report INFSYS RR-1843-02-08, Institut f¨ur Informationssysteme, TU Wien, 2002. Available from `http://www.kr.tuwien.ac.at/research/reports/rr0208.pdf`.

[7] P. A. Boncz, A. N. Wilschut, and M. L. Kersten. Flattening an object algebra to provide performance, In *Proceedings ICDE-98*, pp. 568–577. IEEE Computer Society, 1998.

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

[9] V. Brusoni, L. Console, P. Terenziani, and B. Pernici. Extending temporal relational databases to deal with imprecise and qualitative temporal information. In *Recent Advances in Temporal Databases*, pp. 3–22. Springer, 1995.

[10] J. B. Caouette, E. I. Altman, and P. Narayanan. *Managing Credit Risk: The Next Great Financial Challenge*. John Wiley, 1998.

[11] Y. F. Day, S. Dagtas, M. Iino, A. A. Khokhar, and A. Ghafoor. An object-oriented conceptual modeling of video data. In *Proceedings ICDE-95*, pp. 401–408. IEEE Computer Society, 1995.

[12] Y. F. Day, S. Dagtas, M. Iino, A. A. Khokhar, and A. Ghafoor. Spatio-temporal modeling of video data for on-line object-oriented query processing. In *Proceedings ICMCS-95*, pp. 98–105, 1995.

[13] A. Dekhtyar, R. Ross, and V. S. Subrahmanian. Probabilistic temporal databases, I: Algebra. *ACM Transactions on Database Systems*, 26(1):41–95, 2001.

[14] D. J. Dewitt, N. Kabra, J. Luo, J. Patel, and J. Yu. Client-server paradise. In *Proceedings VLDB-94*, pp. 558–569. Morgan Kaufmann, 1994.

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

[16] D. Dubois and H. Prade. Processing fuzzy temporal knowledge. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(4):729–744, 1989.

[17] S. Dutta. Generalized events in temporal databases. In *Proceedings ICDE-89*, pp. 118–126. IEEE Computer Society, 1989.

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

[19] T. Eiter, J. J. Lu, T. Lukasiewicz, and V. S. Subrahmanian. Probabilistic object bases. *ACM Transactions on Database Systems*, 26(3):264–312, 2001.

[20] T. Eiter, T. Lukasiewicz, and M. Walter. A data model and algebra for probabilistic complex values. *Annals of Mathematics and Artifi cial Intelligence*, 33(2-4):205–252, 2001.

[21] R. Fagin, J. Y. Halpern, and N. Megiddo. A logic for reasoning about probabilities. *Information and Computation*, 87(1-2):78–128, 1990.

[22] S. Gadia, S. Nair, and Y. C. Poon. Incomplete information in relational temporal databases. In *Proceedings VLDB-92*, pp. 395–406. Morgan Kaufmann, 1992.

[23] I. Goralwalla, Y. Leontiev, M. T. Özsu, D. Szafron, and C. Combi. Temporal granularity: Completing the puzzle. *Journal of Intelligent Information Systems*, 16(1):41–63, 2001.

[24] C. S. Jensen and R. T. Snodgrass. Temporal data management. *IEEE Transactions on Knowledge and Data Engineering*, 11(1):36–44, 1999.

[25] S. Khoshafian. *The Jasmine Object-Oriented Database: Multimedia Applications for the Web*. Morgan Kaufman, 1998.

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

[27] Y. Kornatzky and S. E. Shimony. A probabilistic spatial data model. *Information Sciences*, 90:51–74, 1996.

[28] M. Koubarakis. Complexity results for first-order theories of temporal constraints. In *Proceeding KR-94*, pp. 379–390. Morgan Kaufmann, 1994.

[29] M. Koubarakis. Database models for infinite and indefinite temporal information. *Information Systems*, 19(2):141–173, 1994.

[30] S. Kraus, Y. Sagiv, and V. S. Subrahmanian. Representing and integrating multiple calendars. Technical Report CS-TR-3751, University of Maryland, 1996.

[31] H. E. Kyburg, Jr. Interval-valued probabilities. In G. de Cooman, P. Walley, and F. G. Cozman, editors, *Imprecise Probabilities Project*. 1998. Available from `http://ippserv.rug.ac.be/`.

[32] L. V. S. Lakshmanan, N. Leone, R. Ross, and V. S. Subrahmanian. ProbView: A flexible probabilistic database system. *ACM Transactions on Database Systems*, 22(3):419–469, 1997.

[33] L. V. S. Lakshmanan and F. Sadri. Modeling uncertainty in deductive databases. In *Proceedings DEXA-94*, pp. 724–733. *LNCS* 856, Springer, 1994.

[34] L. V. S. Lakshmanan and N. Shiri. A parametric approach to deductive databases with uncertainty. *IEEE Transactions on Knowledge and Data Engineering*, 13(4):554–570, 2001.

[35] G. Özsoyoglu and R. T. Snodgrass. Temporal and real-time databases: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 7(4):513–532, 1995.

[36] S. Ross. *A First Course in Probability*. Prentice Hall, 2001.

[37] H.-J. Schek and P. Pistor. Data structures for an integrated data base management and information retrieval system. In *Proceedings VLDB-82*, pp. 197–207. Morgan Kaufmann, 1982.

[38] G. M. Shaw and S. B. Zdonik. A query algebra for object-oriented databases. In *Proceedings ICDE-90*, pp. 154–162. IEEE Computer Society, 1990.

[39] N. Shiri. *On a Generalized Theory of Deductive Databases*. PhD thesis, Concordia University, Montreal, Canada, August 1997.

[40] R. T. Snodgrass. *Monitoring Distributed Systems: A Relational Approach*. PhD thesis, Carnegie Mellon University, 1982.

[41] R. T. Snodgrass. Temporal object-oriented databases: A critical comparison. In W. Kim, editor, *Modern Database Systems: The Object Model, Interoperability and Beyond*. ACM Press/Addison-Wesley, 1995.

[42] B. Subramanian, T. W. Leung, S. L. Vendenberg, and S. B. Zdonik. The AQUA approach to querying lists and trees in object-oriented databases. In *Proceedings ICDE-95*, pp. 80–89. IEEE Computer Society, 1995.

[43] D. Suciu and J. Paredaens. Any algorithm in the complex object algebra with powerset needs exponential space to compute transitive closure. In *Proceedings PODS-94*, pp. 201–209. ACM Press, 1994.

[44] A. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. T. Snodgrass, editors. *Temporal Databases: Theory, Design, and Implementation*. Benjamin/Cummings, 1994.

[45] S. L. Vendenberg and D. J. Dewitt. Algebraic support for complex objects with arrays, identity and inheritance. In *Proceedings SIGMOD-91*, pp. 158–167. ACM Press, 1991.

Biographies

Veronica Biazzo received the Laurea degree in applied mathematics from the University of Catania, Italy, and the PhD degree in applied mathematics and computer science from the University of Napoly. Currently, she is an assistant professor on the faculty of Science at the University of Catania. She was a visiting researcher at the Computer Science Department of the University of Maryland, College Park. Her area of research includes probability theory and its application in the fi eld of Artifi cial Intelligence.

Rosalba Giugno received the BS degree in computer science from Catania University, Italy, in 1998 and the PhD degree in computer science from Catania University in 2003. She is currently a post doc student in the Department of Computer Science at Catania University. She has been a visiting student at the University of Maryland, College Park, and New York University. Her research

interests include temporal and probabilistic databases, structured databases, image representation, information retrieval, and graph theory.

Thomas Lukasiewicz received the Dipl.-Inf. degree in 1993 from the Technical University of Clausthal, Germany, the Doctorate in 1996 from the University of Augsburg, Germany, and the Dozent degree in 2001 from the Technical University of Vienna, Austria. From 1993 to 1996, he was research assistant at the Institute of Computer Science of the University of Augsburg. From 1997 to 1999, he was assistant professor at the Institute of Computer Science of the University of Gießen. From 1999 to 2001, he was holding a habilitation fellowship by the German Research Foundation (DFG) at the Institute of Information Systems of the Technical University of Vienna. He is currently holding a fellowship by the European Community at the Department of Computer Science of the University of Rome "La Sapienza". His main research interests are databases, knowledge representation and reasoning, and computational complexity. He received the IJCAI-2001 Distinguished Paper Award (joint with Thomas Eiter).

V.S. Subrahmanian received his PhD in Computer Science from Syracuse University in 1989. Since then, he has been on the faculty of the Computer Science Department at the University of Maryland, College Park, where he currently holds the rank of Professor. He received the NSF Young Investigator Award in 1993 and the Distinguished Young Scientist Award from the Maryland Academy of Science in 1997. Prof. Subrahmanian is recognized for his work on nonmonotonic and probabilistic logics, heterogeneous databases, multimedia databases, and probabilistic databases. More recently, he has developed techniques to "agentize" legacy software, allowing multiple software modules to dynamically collaborate with each other to solve complex problems. He has edited two books, one on nonmonotonic reasoning (MIT Press) and one on multimedia databases (Springer). He has co-authored an advanced database textbook (Morgan Kaufman, 1997), a multimedia database textbook, and a monograph on heterogeneous software agents. Prof. Subrahmanian has given invited talks at numerous national and international conferences — in addition, he has served on numerous conference and funding panels, as well as on the program committees of numerous conferences. He has also chaired several conferences. Prof. Subrahmanian is or has previously been on the editorial board of IEEE Transactions on Knowledge and Data Engineering, Artifi cial Intelligence Communications, Multimedia Tools and Applications, Journal of Logic Programming, Annals of Mathematics and Artifi cial Intelligence, Distributed and Parallel Database Journal, and Theory and Practice of Logic Programming.

Prof. Subrahmanian has served on DARPA's (Defense Advanced Research Projects Agency) Executive Advisory Council on Advanced Logistics and as an ad-hoc member of the Air Force Science Advisory Board (2001).