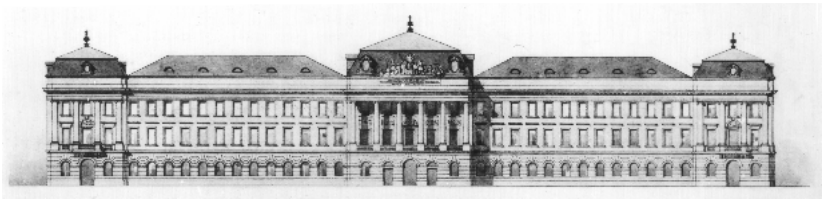
I N F S Y S

R E S E A R C H

R E P O R T



INSTITUT FÜR INFORMATIONSSYSTEME

ABTEILUNG WISSENSBASIERTE SYSTEME

TEMPORAL PROBABILISTIC OBJECT BASES

Veronica BIAZZO Rosalba GIUGNO

Thomas LUKASIEWICZ V.S. SUBRAHMANIAN

INFSYS RESEARCH REPORT 1843-02-08

JUNE 2002

Institut für Informationssysteme
Abtg. Wissensbasierte Systeme
Technische Universität Wien
Favoritenstraße 9-11
A-1040 Wien, Austria
Tel: +43-1-58801-18405
Fax: +43-1-58801-18493
sek@kr.tuwien.ac.at
www.kr.tuwien.ac.at

TECHNISCHE UNIVERSITÄT WIEN

INFSYS RESEARCH REPORT

INFSYS RESEARCH REPORT 1843-02-08, JUNE 2002

TEMPORAL PROBABILISTIC OBJECT BASES

Veronica Biazzo[1] Rosalba Giugno[1]

Thomas Lukasiewicz[2] V.S. Subrahmanian[3]

Abstract. There are numerous applications where we have to deal with temporal uncertainty associated with objects. For example, financial prediction programs often use complex object models and are parameterized by time and uncertainty (e.g., when will a given loan default?). Likewise, in transportation logistics, object models are used to describe objects involved in the transportation process (e.g., vehicles and shipments), and detailed probability distributions exist on shipping and delay times. Thus, in such applications, the ability to automatically store and manipulate time, probabilities, and objects is important. In this paper, we propose a data model and algebra for temporal probabilistic object bases. The data model allows us to associate with each event e a set of possible time points T, and with each time point $t \in T$, an interval for the probability that e occurred at t. We distinguish between explicit object base instances, where the sets of time points along with their probability intervals are simply 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. We define the algebraic operations of selection, restricted selection, renaming, projection, extraction, natural join, Cartesian product, conditional join, and the set operations of intersection, union, and difference on both explicit and implicit object base instances. We show that each operation on implicit object base instances correctly implements its counterpart on explicit object base instances.

---

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

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

[3]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.

Acknowledgements: This work was partially supported by the Army Research Lab under contract number DAAL0197K0135, the Army Research Office under grant number DAAD190010484, by DARPA/RL contract number F306029910552, by the ARL CTA on Advanced Decision Architectures, by NSF grant 937756, by the Austrian Science Fund Project N Z29-INF, by a DFG grant, and by a Marie Curie Individual Fellowship of the European Community (Disclaimer: The authors are solely responsible for information communicated and the European Commission is not responsible for any views or results expressed).

Contents

1 Introduction

Object data models [1, 5, 36, 4] have been used to model numerous applications ranging from multimedia applications[11, 12], financial risk applications[10], and 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.

- We first consider a transportation logistics application [8, 2]. A commercial package delivery company (such as UPS, Fedex, DHL, etc.) has detailed statistical information 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 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" (here, probabilities are degrees of belief about a future event, which may be derived from statistical information about similar previous events). Such an answer is far more helpful to the company's decision making processes than the bland answer "It will be delivered sometime today between 9 am and 5pm". For example, it helps them schedule personnel, prepare receiving facilities (for hazardous material and other heavy 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.
- Weather database systems (such as the Total Atmospheric and Ocean System or TAOS system developed by the US Department of Defense) use object models to represent weather data. Time and uncertainty are omnipresent in weather models and most decision making programs rely on knowledge of this kind of uncertainty in order to make decisions.
- There are numerous financial models [3, 10], which banks and institution lenders use to attempt 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.

In this paper, we propose a theoretical foundation for object bases that allow for representing both temporal data and probabilistic uncertainty and thus in particular also 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 present paper can be summarized as follows:

- We define the concept of a temporal probabilistic object base (TPOB for short). In particular, we introduce the notion of a TPOB-schema, and we define the important concept of (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, renaming, projection, extraction (a new operation), natural join, Cartesian product, conditional join, and the set operations of intersection, union, and difference. We remark that the operations of projection and Cartesian product are simply extensions of their counterparts from the classical relational algebra, while all the 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.
- The TPOB-instances described above are *explicit* in the following sense. Given an event e, for each time point t, an explicit TPOB-instance $\mathbf{I}$ specifies a probability interval[1] describing the probability with which event e occurred (or will occur) at time t. Though explicit TPOB-instances make it easy to formally define the various algebraic operations, they can sometimes be very inefficient from a space point of view (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 $9 \times 60 \times 60 = 32,400$ seconds between 8am and 5pm. In order to avoid this problem, we define the concept of an implicit TPOB-instance. Implicit TPOB-instances represent a probabilistic statement over a set of time points by a probability distribution function in combination with a constraint, which specifies the set of time points. Hence, implicit TPOB-instances allow for an efficient implementation of algebraic operations, while explicit TPOB-instances make defining algebraic operations relatively transparent.
- We show that there is a translation ε that maps an implicit TPOB-instance $\mathbf{I}$ to an explicit one $\varepsilon(\mathbf{I})$. For each algebraic operation on explicit TPOB-instances, we define a counterpart on implicit TPOB-instances. As they work on succinct representations, they have better computational properties.
- We show that the algebraic operations on implicit TPOB-instances correctly implement their counterparts on explicit TPOB-instances (that is, the answers produced by the operations on implicit TPOB-instances succinctly represent the answers produced by the operations 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). For unary algebraic operators op, the correctness theorems are of the form $\varepsilon(op^i(\mathbf{I})) = op^e(\varepsilon(\mathbf{I}))$. We use op^i (resp. op^e) to denote the implicit (resp. explicit) versions of the operator op. A similar kind of correctness result holds (and is shown in the paper) for binary algebraic 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, respectively. Sections 8 and 9 define implicit TPOB-instances and the algebraic

[1] As in the case of [31, 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. [30]). Third, even if we know a point probability for two events e_1 and e_2, in general, we can only infer a probability interval [7] for the conjunction and disjunction of e_1 and e_2 (unless we make additional assumptions such as independence).

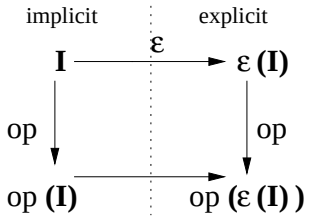


Figure 1: Correctness theorems: commutativity diagram

operations on implicit TPOB-instances, respectively. Section 10 contains directions for future work and concludes the paper. Note that the proofs of all results are given in Appendices A and B.

2 Related Work

To our knowledge, this is the first paper on temporal probabilistic object-oriented databases though it derives heavily in inspiration from [18]. There is extensive work in the literature on temporal databases and temporal object-oriented databases (cf. [34, 23, 43, 42]). Probabilistic extensions of relational databases are also well-explored in the literature; see especially [31, 20] for more background and a detailed discussion of recent work on probabilistic relational databases. Recently, more complex data models have been extended by probabilistic uncertainty in a number of papers. In particular, Eiter et al. [20] presented an approach that adds probabilistic uncertainty to complex value relational databases, while Kornatzky and Shimony [25, 26] and Eiter et al. [19] described approaches to probabilistic object-oriented databases. Our approach in this paper is a temporal extension of the model by Eiter et al. [19]. In addition, this paper introduces a new implicit data model and an implicit algebra which correctly implements its explicit counterpart, and which can be more efficiently realized. Moreover, the two operations of restricted selection and extraction are brand new.

There is, however, very little work on the integration of temporal reasoning and probabilistic databases. In particular, Dyreson and Snodgrass in their pioneering work [18] and subsequently Dekhtyar et al. [13] presented approaches to temporal indeterminacy in relational databases based on probabilistic uncertainty. Dyreson and Snodgrass [18] extend the *SQL data model and query language* by probabilistic uncertainty on time points. They add indeterminate temporal attributes (which have indeterminate instants as associated values) to SQL. Indeterminate instants are intervals of time points with associated probability distributions. The SQL query language is extended by a construct to define *ordering plausibility* which is an integer between 1 and 100 that specifies to which degree the result of an SQL query should contain uncertain answers (where 1 means that any possible answer to a query is desired, while 100 says that only definite answers to a query are desired). Moreover, there is a construct to define the *correlation credibility*, which specifies simple modifications of the probability distributions in the base relations before evaluating the selection condition in SQL queries. Dyreson and Snodgrass also describe efficient data structures and query processing algorithms for their approach. Dekhtyar et al. [13], in contrast, extend the *relational data model and algebra* by temporal indeterminacy based on probabilities. 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.

There is further work on nonprobabilistic temporal indeterminacy in databases. In particular, Snodgrass [39] models indeterminacy using a model that is based on a three-valued logic. Moreover, Dutta [17] and Dubois and Prade [16] propose a fuzzy logic approach to temporal indeterminacy, while Koubarakis [28, 27] and Brusoni et al. [9] suggest approaches based on constraints. Finally, Gadia et al. [21] introduce partial temporal databases, which are based on partial temporal elements.

Our work in this paper is perhaps 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, however, 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].

The work by Dyreson and Snodgrass [18] is also closely related, but differs from ours in several ways. First, we present an extension of object-oriented databases, while their approach is an extension of relational databases. 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 probabilistically independent from each other. 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 their work allows only for precise point probabilities over intervals of time points.

Our work is also related to data models and algebraic operations for complex objects [1, 37, 44, 41, 40, 6]. Our work is a strict extension of the algebra for complex values presented by Abiteboul et al. [1]. As in the case of Shaw and Zdonik [37], Vandenberg and DeWitt [44], and Boncz et al. [6], our data model supports the type constructors for sets and tuples on elementary datatypes. Like them, we also support the algebraic operations of selection, projection, join, union, intersection, and difference. However, unlike them, we do not support the type constructors for arrays and multisets as in [44], user-defined abstract datatypes as in [37], and classes as elementary datatypes as in [6], and aggregate operations and next/unnest operations. Of course, our work involves time and probabilities that are not considered in these papers. Extending our work to such types and algebraic operations is an interesting topic of future research. The nested relational algebra described in [41] 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 [41]. Finally, Subramanian et al. [40] 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 in some sense related to our extraction operation, which extracts a subhierarchy from the class hierarchy of a temporal object base.

3 Basic Definitions

We now recapitulate some basic definitions. We first recall the notion of a calendar due to Kraus et al. [29], which serves as a temporal atomic type in our model. We then define types and their values. The set of all values of a type τ is also called the *domain* of τ, denoted $\mathrm{dom}(\tau)$. We first introduce (non-probabilistic) classical types and their values. 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.

3.1 Calendars

We now recall the concept of a calendar due to Kraus et al. [29]. Intuitively, a calendar consists of a finite sequence of time units and a predicate specifying a set of valid time points over this sequence. It is used as an elementary temporal type with the set of all valid time points as associated domain of values.

We first define time units and linear temporal hierarchies, which are essentially finite sequences of time units. 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 [29]. 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. □

3.2 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, which are used to reference components of values of tuple types (in a similar way as attributes in relational database schemas are used to reference fields of tuples).

Classical types are either atomic types or complex types constructed from atomic types and attributes by using the set and the tuple constructor. We formally define *classical types* by induction as follows:

- 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 $[A_1 \colon \tau_1, \ldots, A_k \colon \tau_k]$ is a classical type (called *classical tuple type*).

We give some examples of classical types.

Example 3.3 Consider an application maintaining information about how long it takes for packages to get from one location to another. Such an application may be used by a package delivery service like DHL, Fedex, or UPS. The attributes Origin and Destination may be defined over the classical atomic type string, while the attribute Contents may be defined over the classical set type {string}. A classical tuple type is [Origin: string, Destination: string, Contents: {string}]. □

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

- For all classical atomic types $\tau \in \mathcal{T}$, every $v \in \text{dom}(\tau)$ 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.

3.3 Probabilistic Types

Probabilistic types are used to 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. We formally define *probabilistic types* by induction as follows (observe that the set of all probabilistic types includes all classical tuple types):

- 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.*

The following example illustrates the concept of a probabilistic type.

Example 3.5 In the Package Example, the attributes Delivery and STOPone may be defined over the atomic probabilistic type [[time]] and the probabilistic tuple type [City: string, Arrive: string, Shipment: [[time]]], respectively, where time is a calendar. □

The values of probabilistic types are appropriately typed random variables. We define *values* of probabilistic types by induction as follows:

- 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 $[A_1 : v_1, \ldots, A_k : v_k]$, where $v_1, \ldots, v_k$ are values of $\tau_1, \ldots, \tau_k$. Given a value $v = [A_1 : v_1, \ldots, A_k : v_k]$, 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 from the set $\{v_1, \ldots, v_n\}$. 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 $hour \sqsupseteq minute$. An value of the atomic probabilistic type [[time]] is $\{((12, 30), [.4, .6]), ((12, 35), [.4, .6])\}$. Intuitively it says that this value is either $(12, 30)$ or $(12, 35)$. The probability that the value is $(12, 30)$ is 0.4–0.6 and similarly for $(12, 35)$. Similarly, an explicit value for the atomic probabilistic type [[string]] is {(New_York, $[.3, .4]$), (Washington, $[.5, .6]$)}. □

Notice that 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, if $v = \{(5, [1, 1]), (6, [1, 1])\}$ then this is inconsistent (as it says that v equals both 5 and 6 with probability 1 which is impossible. Thus, we say that v is *consistent* iff there exists a probability function Pr (cf. [35]) on $\{v_1, \ldots, v_n\}$ (that is, a mapping $Pr\colon \{v_1, \ldots, v_n\} \to [0, 1]$ 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 not difficult to see that such a Pr exists iff $l_1 + \cdots + l_n \leq 1 \leq u_1 + \cdots + u_n$.

We can extend the above definition naturally to obtain the following notion of consistency for values of probabilistic types. A value $v = \{(v_1, [l_1, u_1]), \ldots, (v_n, [l_n, u_n])\}$ of an atomic probabilistic type 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.

3.4 Probabilistic Strategies

We now define the concept of a probabilistic conjunction (resp., disjunction, difference) strategy, which is used in our algebraic operations to compute the probability interval of a conjunction (resp., disjunction, difference) of two pieces of information, which are each associated with a given probability interval.

Consider two events e_1 and e_2, which have 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 $e_1 \wedge e_2$, $e_1 \vee e_2$, and $e_1 \wedge \neg e_2$, 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 $e_1 \wedge e_2$, $e_1 \vee e_2$, and $e_1 \wedge \neg e_2$. More formally, let $\mathcal{U}$ be the set of all nonempty subintervals $[l, u]$ of the unit interval $[0, 1]$. Assume that the probabilities of the events e_1 and e_2 are in the intervals $[l_1, u_1]$ and $[l_2, u_2]$, respectively. A *conjunction* (resp., *disjunction, difference*) *strategy* is a function $\otimes\colon \mathcal{U}^2 \to \mathcal{U}$ (resp., $\oplus\colon \mathcal{U}^2 \to \mathcal{U}$, $\ominus\colon \mathcal{U}^2 \to \mathcal{U}$) that computes the probability interval of $e_1 \wedge e_2$ (resp., $e_1 \vee e_2$, $e_1 \wedge \neg e_2$) for some fixed dependencies between e_1 and e_2 (or lack thereof).

Lakshmanan et al. [31] give axioms that conjunction and disjunction strategies should satisfy, but we do not repeat these here, except to say that our conjunction and disjunction strategies should also satisfy such axioms. Tables 1 and 2 show some examples of conjunction, disjunction, and difference strategies (see [20] for more examples). For associative and commutative conjunction (resp., disjunction) strategies $\odot$ and nonempty intervals $[l_1, u_1], \ldots, [l_k, u_k] \subseteq [0, 1]$ with $k \geq 1$, we use $\bigodot_{i=1}^{k}[l_i, u_i]$ to denote $[l_1, u_1] \odot \cdots \odot [l_k, u_k]$. For $k = 0$, we define $\bigodot_{i=1}^{k}[l_i, u_i]$ as the constants $[1, 1]$ (resp., $[0, 0]$).

4 Temporal Probabilistic Object Bases

In this section, we first introduce the concept of a schema for temporal probabilistic object bases. Intuitively, a schema consists of two parts. The first is a hierarchy of classes with associated types. The second part specifies 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." In this case, we may have a probability of 0.05 that an arbitrary letter is a registered letter.

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 temporal probabilistic object base instances with respect to this inheritance completion schema.

Table 1: Conjunction strategies

Mutual exclusion	$([l_1,u_1] \otimes_{me} [l_2,u_2]) = [0,0]$
Positive correlation	$([l_1,u_1] \otimes_{pc} [l_2,u_2]) = [\min(l_1,l_2), \min(u_1,u_2)]$
Independence	$([l_1,u_1] \otimes_{in} [l_2,u_2]) = [l_1 \cdot l_2,\ u_1 \cdot u_2]$
Ignorance	$([l_1,u_1] \otimes_{ig} [l_2,u_2]) = [\max(l_1,l_2), \min(1,u_1+u_2)]$

Table 2: Disjunction strategies

Mutual exclusion	$([l_1,u_1] \oplus_{me} [l_2,u_2]) = [\min(1,l_1+l_2), \min(1,u_1+u_2)]$
Positive correlation	$([l_1,u_1] \oplus_{pc} [l_2,u_2]) = [\max(l_1,l_2), \max(u_1,u_2)]$
Independence	$([l_1,u_1] \oplus_{in} [l_2,u_2]) = [l_1+l_2-l_1 \cdot l_2,\ u_1+u_2-u_1 \cdot u_2]$
Ignorance	$([l_1,u_1] \oplus_{ig} [l_2,u_2]) = [\max(0,l_1+l_2-1), \min(u_1,u_2)]$

Table 3: Difference strategies

Mutual exclusion	$([l_1,u_1] \ominus_{me} [l_2,u_2]) = [l_1, \min(u_1,1-l_2)]$
Positive correlation	$([l_1,u_1] \ominus_{pc} [l_2,u_2]) = [\max(0,l_1-u_2), \max(0,u_1-l_2)]$
Independence	$([l_1,u_1] \ominus_{in} [l_2,u_2]) = [l_1 \cdot (1-u_2),\ u_1 \cdot (1-l_2)]$
Ignorance	$([l_1,u_1] \ominus_{ig} [l_2,u_2]) = [\max(0,l_1-u_2), \min(u_1,1-l_2)]$

4.1 Temporal Probabilistic Object Base Schema

We now define the concept of a temporal probabilistic object base schema. Informally, every schema specifies 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 class $c \in \mathcal{C}$ 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 4 below;
- $\mathrm{me}(\mathsf{Package}) = \{\{\mathsf{Letter}, \mathsf{Box}, \mathsf{Tube}\}, \{\mathsf{Priority}, \mathsf{Express\_saves}\}\}$,
  $\mathrm{me}(\mathsf{Box}) = \mathrm{me}(\mathsf{Express\_saves}) = \{\{\mathsf{Two\text{-}transfer}\}\}$,
  $\mathrm{me}(\mathsf{Tube}) = \mathrm{me}(\mathsf{Priority}) = \{\{\mathsf{One\text{-}transfer}\}\}$,
  $\mathrm{me}(\mathsf{Letter}) = \mathrm{me}(\mathsf{One\text{-}transfer}) = \mathrm{me}(\mathsf{Two\text{-}transfer}) = \emptyset$;

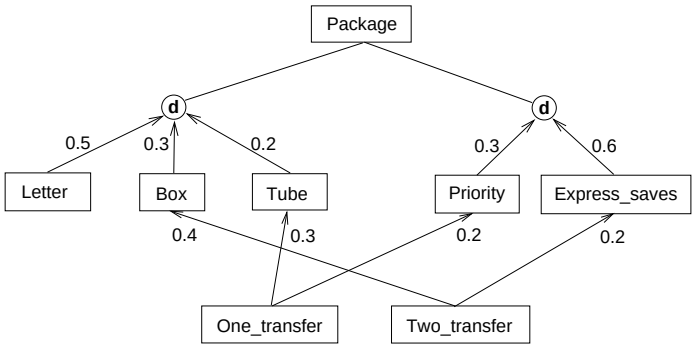


Figure 2: Package Example with probability assignment $\wp$

- the probability assignment $\wp$ is given in Fig. 2.

Note that the acyclic directed graph $(\mathcal{C}, \Rightarrow)$ is the graph resulting from Fig. 2 when the d-nodes are contracted to Package and the probability labels are removed. The d-nodes of Fig. 2 denote disjoint decompositions – every package is either a letter, box, or tube, and either an express package or a priority package. □

Table 4: 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]]]]

We now introduce some concepts related to TPOB-schemas. 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 3.3) 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 $k \geq 1$. We denote by $\Rightarrow^\star$ the reflexive and transitive closure of $\Rightarrow$. We say c_1 is a *subclass* (resp., *strict subclass*) of c_2, or c_2 is a *superclass* (resp., *strict superclass*) of 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.

We finally define the important notion of consistency of TPOB-schemas. 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. Every TPOB-schema should have this property. 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 $\mathcal{O}$. The interpretation $\mathcal{I}$ is a *taxonomic model* of $\mathbf{S}$ iff the following conditions are satisfied:

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 Figure 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).

The interpretation $\mathcal{I}$ 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 $c \Rightarrow d$, axiom (C4) below holds:

C4 $|\zeta(c)| = \wp(c, d) \cdot |\zeta(d)|$.

We say $\mathbf{S}$ 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 axiom above 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.

Note that the work in this section builds upon definitions in [19]. There, it is shown that deciding the consistency of probabilistic object base schemas is NP-complete — this result also applies here. However, there are important special cases of TPOB-schemas, which can be tested in polynomial time, and for which deciding consistency can also be done in polynomial time (see [19] for detailed algorithms).

4.2 Inheritance Completion

The inheritance completion of a TPOB-schema adds to the type of every class $c \in \mathcal{C}$, all the attributes (with their types) that are inherited from all superclasses of c. We use inheritance strategies to specify how to resolve conflicts that arise due to multiple inheritance. More precisely, we add to the type of each class $c \in \mathcal{C}$, every top-level attribute A with its type at a minimal superclass of c. If more than one such minimal superclass exists, then we have a conflict due to multiple inheritance, and we use an inheritance strategy to select a unique class among the set of all such minimal superclasses.

More 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 5. □

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

Table 5: 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). Note that we assume that $\mathcal{O}$ is countably infinite to ensure that we may have arbitrary large finite TPOB-instances. For the algebraic operations of natural join, Cartesian product, and conditional join (see Section 6), we additionally assume that $\mathcal{O}$ is closed under Cartesian product, that is, $\mathcal{O} \times \mathcal{O} \subseteq \mathcal{O}$.

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 oids 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 6 and 7 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%. □

Table 6: 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	$\{o_5\}$	$\{o_5\}$

We now define the concept of a probabilistic extent of a class. In classical object bases, the extent of a class c is a mapping $\mathrm{ext(c)}$ that associates with every object o in the object base the number 0 (resp., 1)

Table 7: 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: $\{((18,00),[.3,.5]),((18,30),[.6,.7])\}$]]

iff o does not belong (resp., does belong) to the class c. In probabilistic object bases, the probabilistic extent of a class 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 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. More 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. This is important: intuitively, given an object o and a class c, there must be only one probability that o belongs to c (otherwise something would be wrong). For example, given that o is a package in Figure 2, we know that the probability of its having one transfer is 0.06 — notice that Figure 2 has two paths and the computed probability across *both* paths is 0.06. This is the kind of intuition that the notion of *coherence* below attempts to capture. 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.

Another important concept for TPOB-instances is that of consistency, which is simply an extension of the concept of consistency for values of probabilistic types. A TPOB-instance is consistent iff all its values of probabilistic types are consistent. Inconsistent values of probabilistic types and inconsistent TPOB-instances are meaningless. Hence, the notion of consistency is very important. Formally, 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 the unary operations of the TPOB-algebra. These operations take a TPOB-instance over a TPOB-schema as input, and produce a TPOB-instance over a TPOB-schema as output. We

describe the unary operations of selection, restricted selection, renaming, projection, and extraction. *Unless specified otherwise, we assume that all input TPOB-schemas are fully inherited.*

5.1 Selection

The selection condition finds all objects in a TPOB-instance $\mathbf{I}$ which satisfy a probabilistic selection condition ϕ. We will formally define a probabilistic selection condition in this section. A simple probabilistic selection condition says that ordinary (i.e., non-probabilistic) selection conditions must be true with probability inside a given range. Boolean combinations of such simple probabilistic selection conditions are also allowed. A selection condition is a logical combination of atomic selection conditions. An atomic selection condition may include membership of an object in a class membership, or a comparison that involves attribute values of an object. Path expressions (defined below) refer to "inner" attribute values of an object.

Hence, we first define path expressions and atomic selection conditions, which are then used to construct selection conditions and probabilistic selection conditions. In the rest of this section, let $\mathbf{I} = (\pi, \nu)$ be 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. We formally define *path expressions* by induction as follows. A path expression for the tuple type $[A_1 : \tau_1, \ldots, A_k : \tau_k]$ 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. The following shows some examples of path expressions.

Example 5.1 STOPone, STOPone.City, and STOPone.Arrive.[[time]] 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 $v = [A_1 : v_1, \ldots, A_k : v_k]$ 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 $v = \{(v_1, I_1), \ldots, (v_k, I_k)\}$ 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

An atomic selection condition describes either (i) a class membership of an object, or (ii) a comparison between an attribute value of an object and a constant value, or (ii) a comparison between two attribute values of an object. More formally, an *atomic selection condition* 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 $\theta \in \{\leq, <, =, \neq, >, \geq\}$.

We give some examples of atomic selection conditions in the Package Example.

Example 5.3 $in(\mathsf{Letter})$ is an atomic selection condition which specifies the set of all objects that are letters. Likewise, $\mathsf{STOPone.City} = \mathsf{New\_York}$ is an atomic selection condition which specifies the set of "all objects that have New York as the first stop." $\mathsf{Origin} =_{\otimes in} \mathsf{Destination}$ is an atomic selection condition which specifies all objects whose Origin and Destination are the same, assuming the value of the Origin is independent of the value of the Destination. □

We now define the meaning of an atomic selection condition 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. More 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 $\mathsf{STOPone.City{=}New\_York}$ 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. A formal inductive definition is as follows. (i) Every atomic selection condition 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. The following shows an example of a selection condition from the Package Example.

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 that 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 "well-formed" 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:

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

The following example shows some probabilistic valuations of selection conditions.

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 $\text{prob}_{\mathbf{I},o_3}$ and $\text{prob}_{\mathbf{I},o_5}$, respectively. □

5.1.4 Probabilistic Selection Conditions

We define *probabilistic selection conditions* inductively as follows. 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 probabilistic selection condition (an *atomic* one). If ϕ and ψ are probabilistic selection conditions, 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.

Intuitively, a probabilistic selection condition of the form $(\phi)[l, u]$ says find all objects that satisfy condition ϕ with probability between l and u (both inclusive).

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 $\text{prob}_{\mathbf{I},o} \models \phi$, is inductively defined by:

- $\text{prob}_{\mathbf{I},o} \models (\phi)[l, u]$ iff $\text{prob}_{\mathbf{I},o}(\phi) \subseteq [l, u]$;
- $\text{prob}_{\mathbf{I},o} \models \neg\phi$ 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 $c \in \mathcal{C}$.
- $\nu' = \nu \,|\, \pi'(\mathcal{C})$.

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 8. □

Table 8: 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

Restricted selection is a new operator not present in the usual relational (or object) algebra. Consider an atomic selection condition ϕ of the form $P\,\theta\,w$. Informally, this operation is a selection with respect to ϕ on the value $v = \{(v_1, I_1), \ldots, (v_k, I_k)\}$ of an atomic probabilistic type "inside" the value $\nu(o)$ of every object o. As usual, the path expression P in ϕ is used to specify the part v of the value $\nu(o)$.

We first formally define the restricted selection operation on values of probabilistic tuple types as follows. 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 atomic selection condition of the form $P\,\theta\,w$, where P is a path expression for τ. Then, 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 $o \in \pi'(\mathcal{C})$.

The following example illustrates the use of the restricted selection operator.

Example 5.10 Consider the TPOB-instance $\mathbf{I} = (\pi, \nu)$ of Example 5.9 and the atomic selection condition $\phi = (\mathsf{Time} < (8, 05))$. Then, the restricted selection on $\mathbf{I}$ with respect to ϕ is given by the TPOB-instance $\sigma^r_\phi(\mathbf{I}) = (\pi', \nu')$, where $\pi' = \pi$ and ν' is shown in Table 9. □

Table 9: ν' resulting from restricted selection

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

5.3 Renaming

We now define the renaming operation. Informally, this operation renames some attributes in types of TPOB-schemas and in values of TPOB-instances. We use path expressions to allow for a renaming of attributes at lower levels inside types and values. We first define the syntax of renaming conditions, which specify which attributes are to be renamed, and how they are to be renamed.

A *renaming condition* for a probabilistic tuple type τ is an expression of the form $\vec{\mathbf{P}} \leftarrow \vec{\mathbf{Q}}$, where $\vec{\mathbf{P}} = P_1, \ldots, P_l$ is a list of pairwise distinct path expressions for τ, and $\vec{\mathbf{Q}} = Q_1, \ldots, Q_l$ is a list of pairwise distinct path expressions such that P_i and Q_i differ exactly in their rightmost attribute, for every $i \in \{1, \ldots, l\}$. We illustrate the concept of renaming conditions via our Package Example.

Example 5.11 Let τ be the probabilistic tuple type [STOPone: [City: string, Arrive: [[time]], Shipment: [[time]]]]. A renaming condition for τ is STOPone.City, STOPone←STOPone.City1, STOPone1. □

5.3.1 Renaming of TPOB-Schemas

Before defining how to apply the renaming operator on TPOB-instances, we need two definitions — one on applying it to probabilistic tuple types, and another on applying it to TPOB-schemas.

Let N be a renaming condition of the form $P \leftarrow P'$ for the probabilistic tuple type $\tau = [A_1 \colon \tau_1, \ldots, A_n \colon \tau_n]$. The *renaming* of τ with respect to N, denoted $\delta_N(\tau)$, is defined as follows:

- If $P = A_i$ and $P' = A_i{}'$, then $\delta_N(\tau)$ is obtained from τ by replacing A_i by $A_i{}'$.
- If $P = A_i.[\![R]\!]$, $P' = A_i.[\![R']\!]$, and τ_i is an atomic probabilistic type, then $\delta_N(\tau)$ is obtained from τ by replacing $\tau_i = [\![\tau_i{}']\!]$ by $[\![\delta_{R \leftarrow R'}(\tau_i{}')]\!]$.
- If $P = A_i.R$, $P' = A_i.R'$, and τ_i is not an atomic probabilistic type, then $\delta_N(\tau)$ is obtained from τ by replacing τ_i by $\delta_{R \leftarrow R'}(\tau_i)$.

Let $N = P_1, \ldots, P_l \leftarrow P_1{}', \ldots, P_l{}'$ be a renaming condition for τ. The *renaming* of τ with respect to N, denoted $\delta_N(\tau)$, is defined as the simultaneous renaming on τ with respect to all $P_i \leftarrow P_i{}'$.

We now define the renaming of TPOB-schemas as follows. Let $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ be a TPOB-schema and let N be a renaming condition for every $\sigma(c)$ with $c \in \mathcal{C}$. The *renaming* of $\mathbf{S}$ with respect to N, denoted $\delta_N(\mathbf{S})$, is the TPOB-schema $(\mathcal{C}, \sigma', \Rightarrow, \mathrm{me}, \wp)$, where $\sigma'(c) = \delta_N(\sigma(c))$ for all $c \in \mathcal{C}$.

5.3.2 Renaming of TPOB-Instances

Before defining the renaming on TPOB-instances, we need to define it on values of probabilistic tuple types.

Let N be a renaming condition of the form $P \leftarrow P'$ for the probabilistic tuple type $\tau = [A_1 \colon \tau_1, \ldots, A_n \colon \tau_n]$. Let $v = [A_1 \colon v_1, \ldots, A_n \colon v_n]$ be a value of τ. The *renaming* of v w.r.t. N, denoted $\delta_N(v)$, is defined by:

- If $P = A_i$ and $P' = A_i{}'$, then $\delta_N(v)$ is obtained from v by replacing A_i by $A_i{}'$.
- If $P = A_i.[\![R]\!]$, $P' = A_i.[\![R']\!]$, and v_i is a value of an atomic probabilistic type, then $\delta_N(v)$ is obtained from v by replacing every $(v_i{}', I_i) \in v_i$ by $(\delta_{R \leftarrow R'}(v_i{}'), I_i)$.
- If $P = A_i.R$, $P' = A_i.R'$, and v_i is not a value of an atomic probabilistic type, then $\delta_N(v)$ is obtained from v by replacing v_i by $\delta_{R \leftarrow R'}(v_i)$.

Let $N = P_1, \ldots, P_l \leftarrow P_1{}', \ldots, P_l{}'$ be a renaming condition for τ. The *renaming* of v with respect to N, denoted $\delta_N(v)$, is defined as the simultaneous renaming on v with respect to all $P_i \leftarrow P_i{}'$.

We are now ready to define the renaming of TPOB-instances as follows. Let $\mathbf{I} = (\pi, \nu)$ be a TPOB-instance over a TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$, and let N be a renaming condition for every $\sigma(c)$ with $c \in \mathcal{C}$. The *renaming* of $\mathbf{I}$ with respect to N, denoted $\delta_N(\mathbf{I})$, is the TPOB-instance (π, ν') over the TPOB-schema $\delta_N(\mathbf{S})$, where $\nu'(o) = \delta_N(\nu(o))$ for all $o \in \pi(\mathcal{C})$.

5.4 Projection

Intuitively, the projection operation 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. We formally define the projection of a TPOB-schema (resp., TPOB-instance) on a set of top-level attributes as follows.

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 defined as 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 defined as 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 5.12 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 $\mathbf{A} = \{\mathsf{Origin}, \mathsf{Contents}, \mathsf{Time}\}$. 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 10 (resp., 11). □

Table 10: 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 11: 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])}]

5.5 Extraction

The extraction operation allows for the elimination of some classes from the class hierarchy of a TPOB-schema. That is, the extraction operation removes some classes from a TPOB-schema, and all objects in the removed 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 operation on TPOB-schemas and then extend it to TPOB-instances.

5.5.1 Extraction on TPOB-Schemas

Roughly speaking, the extraction operation on a TPOB-schema can be described as follows. Given a TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \mathrm{me}, \wp)$ and a set of classes $\mathbf{C} \subseteq \mathcal{C}$, the extraction on $\mathbf{S}$ with respect to $\mathbf{C}$ produces the TPOB schema $\mathbf{S}' = (\mathbf{C}, \sigma', \mathrm{me}', \wp')$ that is obtained by restricting $\mathbf{S}$ to the classes in $\mathbf{C}$. That is, the new TPOB-schema $\mathbf{S}'$ contains only the classes in $\mathbf{C}$ and their types. Moreover, the mapping me and the probability assignment $\wp$ are adapted to the classes in $\mathbf{C}$ in order to obtain me' and $\wp'$. For example, if we have the immediate subclass relationships $c_1 \Rightarrow c_2 \Rightarrow c_3$ in $\mathbf{S}$ with the associated probabilities $\wp(c_1, c_2)$ and $\wp(c_2, c_3)$, and c_1 and c_3 belong to $\mathbf{C}$ but c_2 does not, then we assume the immediate subclass relationship $c_1 \Rightarrow' c_3$ in $\mathbf{S}'$ with the associated probability $\wp'(c_1, c_3) = \wp(c_1, c_2) \cdot \wp(c_2, c_3)$. Moreover, if a class $c_1 \in \mathbf{C}$ is disjoint to a class $c_2 \notin \mathbf{C}$ in $\mathbf{S}$, then all the extracted subclasses of c_2 are disjoint to c_1 in $\mathbf{S}'$.

More 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 5.13 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 12.
- 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 12: 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]]]

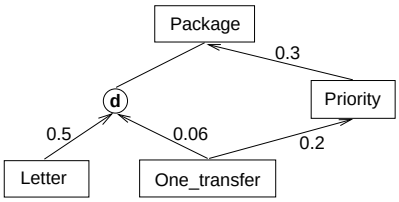


Figure 3: Class hierarchy and probability assignment resulting from extraction

5.5.2 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 $\Xi_{\mathbf{C}}(\mathbf{S}) = (\mathcal{C}', \sigma', \mathrm{me}', \wp')$, where:

- π' is the restriction of π to $\mathcal{C}'$.
- ν' is the restriction of ν to $\pi'(\mathcal{C}')$.

The following example illustrates the extraction on TPOB-instances.

Example 5.14 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 5.13, where π' is given by Table 13 and ν' is given by $\nu'(o_3) = \nu(o_3)$. □

Table 13: 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

In this section, we define algebraic operations to combine information from two TPOB-instances. These operations take two TPOB-instances over two TPOB-schemas as input and produce a new TPOB-instance over a (possibly new) TPOB-schema as output. We consider the binary operations of natural join, Cartesian product, conditional join, and the set operators of intersection, union, and difference. Again, unless specified otherwise, we assume that all input TPOB-schemas are fully inherited.

6.1 Natural Join

Our *natural join* operator is inspired by its counterpart in classical relational databases, which forms a Cartesian product of two TPOB-instances, performs a selection forcing equality on those attributes that

appear in both relation schemes, and finally removes duplicate values.

The main idea behind the natural join operation can be roughly described as follows. Recall that for each class $c \in \mathcal{C}$ of a TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \mathrm{me}, \wp)$, the type $\sigma(c) = [A_1 : \tau_1, \ldots, A_l : \tau_l]$ is a probabilistic tuple type over some top-level attributes $A_1, \ldots, A_l$. Moreover, each object $o \in \pi(c)$ in a TPOB-instance $\mathbf{I} = (\pi, \nu)$ over $\mathbf{S}$ is associated with a value $\nu(o) = [A_1 : v_1, \ldots, A_l : v_l]$ of type $\sigma(c)$. Given two TPOB-instances $\mathbf{I}_1$ and $\mathbf{I}_2$ over the TPOB-schemas $\mathbf{S}_1$ and $\mathbf{S}_2$, respectively, the TPOB-instance $\mathbf{I}$ resulting from the natural join of $\mathbf{I}_1$ and $\mathbf{I}_2$ contains an object $o = (o_1, o_2)$ for certain pairs of objects o_1 and o_2 in $\mathbf{I}_1$ and $\mathbf{I}_2$, respectively. The value $\nu(o)$ associated with each such o is given by the natural join of the values $\nu_1(o_1)$ and $\nu_2(o_2)$ in $\mathbf{I}_1$ and $\mathbf{I}_2$, respectively, which is roughly a concatenation combined with an intersection on common attribute values. The new TPOB-instance $\mathbf{I}$ is defined over a TPOB-schema $\mathbf{S}$ that contains a class $c = (c_1, c_2)$ for any two classes c_1 and c_2 in $\mathbf{S}_1$ and $\mathbf{S}_2$, respectively. The type of each such class c is obtained by merging the types of c_1 and c_2 in $\mathbf{S}_1$ and $\mathbf{S}_2$, respectively, and the new class hierarchy in $\mathbf{S}$ is obtained by merging the two class hierarchies in $\mathbf{S}_1$ and $\mathbf{S}_2$.

We first formalize the natural join of two TPOB-schemas. We then define 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 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 $\mathbf{S}_1$).

More formally, we define the natural join operation on two TPOB-schemas as follows. If the 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 5.13, 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 14 and Figure 4. □

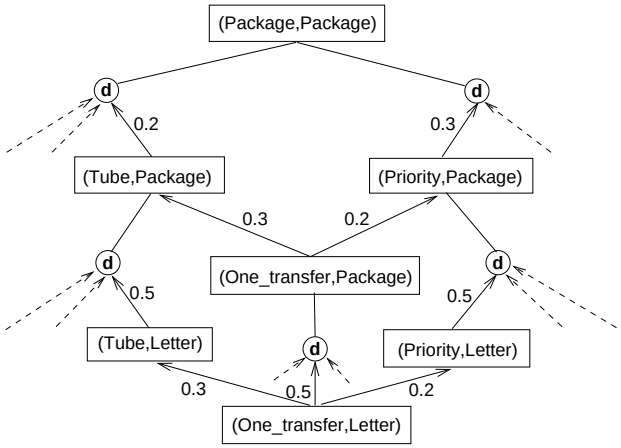


Figure 4: Natural join of schemas

Table 14: 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 now first introduce the intersection of two values v_1 and v_2 of the same classical or probabilistic type τ. The *intersection* of v_1 and v_2 under a conjunction strategy $\otimes$, denoted $v_1 \cap_\otimes v_2$, is inductively defined by:

- 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 $[\![\mathsf{time}]\!]$:

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

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 follows:

- $(v_1 \bowtie_\otimes v_2).A = v_i.A$ 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

We now define the natural join of two TPOB-instances as follows. 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 produced in Example 5.13, 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 5.14, respectively. The natural join of $\mathbf{I}_1$ and $\mathbf{I}_2$ under the conjunction strategy for ignorance $\otimes_{ig}$ 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 15. □

Table 15: 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 and Conditional Join

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.

The conditional join operation combines values of two TPOB-instances that satisfy a probabilistic selection condition ϕ. Let $\mathbf{I}_1$ and $\mathbf{I}_2$ be TPOB-instances over the Cartesian-product-compatible TPOB-schemas $\mathbf{S}_1$ and $\mathbf{S}_2$, respectively. The *conditional join* of $\mathbf{I}_1$ and $\mathbf{I}_2$ with respect to ϕ, denoted $\mathbf{I}_1 \bowtie_\phi \mathbf{I}_2$, is the TPOB-instance $\sigma_\phi(\mathbf{I}_1 \times \mathbf{I}_2)$ over the TPOB-schema $\mathbf{S}_1 \times \mathbf{S}_2$.

Example 6.4 Let $\mathbf{S}_1$ and $\mathbf{I}_1$ be the TPOB-schema and the TPOB-instance, respectively, produced in Example 5.12. Let $\mathbf{S}_2$ and $\mathbf{I}_2$ be the TPOB-schema and the TPOB-instance obtained from $\mathbf{S}_1$ and $\mathbf{I}_1$, respectively, by renaming the attributes Origin, Contents, and Time with Origin1, Contents1, and Time1, respectively. The Cartesian product of $\mathbf{S}_1$ and $\mathbf{S}_2$ is the TPOB-schema $\mathbf{S}_1 \times \mathbf{S}_2 = (\mathcal{C}, \sigma, \mathrm{me}, \wp)$ partially shown in Table 16 and Figure 4. The condition join of $\mathbf{I}_1$ and $\mathbf{I}_2$ with respect to $\phi = (\mathsf{Origin} = \mathsf{Rome} \wedge \mathsf{Origin1} = \mathsf{Paris})[1, 1]$ is the TPOB-instance $\sigma_\phi(\mathbf{I}_1 \times \mathbf{I}_2) = (\pi, \nu)$ over the TPOB-schema $\mathbf{S}_1 \times \mathbf{S}_2$, where $\pi((\mathsf{Priority}, \mathsf{Two\text{-}transfer})) = \{(o_3, o_5)\}$ and $\pi(c) = \emptyset$ for all other $c \in \mathcal{C}$, and ν is shown in Table 17. □

Table 16: Type assignment σ resulting from conditional join

c	$\sigma(c)$
(Package, Package)	[Origin: string, Origin1: string]
(Tube, Package)	[Origin: string, Contents: {string}, Origin1: string]
(Priority, Package)	[Origin: string, Time: [[time]], Origin1: string]
(One-transfer, Letter)	[Origin: string, Contents: {string}, Time: [[time]], Origin1: string]

Table 17: Value assignment ν resulting from conditional join

o	$\nu(o)$
(o_3, o_5)	[Origin: Rome, Time: {((8, 00), [.45, .5]), ((8, 10), [.4, .5])}, Origin1: Paris, Time1: {((12, 00), [.3, .4]), ((12, 05), [.4, .7])}]

6.3 Intersection, Union, and Difference

In this section, we define the set operations of intersection, union, and difference on two TPOB-instances over the same TPOB-schema. Informally, the intersection operation intersects the sets of objects of the two TPOB instances, as well as the two values associated with each object in both TPOB-instances. The union operation, in contrast, computes the union of the sets of objects of the two TPOB-instances, combined with the union of the two values associated with each object in both TPOB-instances. Finally, the difference operation returns the set of objects of the first TPOB-instance, combined with the difference of the two values associated with each object in both TPOB-instances.

We first define the intersection of two TPOB-instances. We then introduce the union of two values and two TPOB-instances, and finally the difference of two values and two TPOB-instances.

6.3.1 Intersection of TPOB-Instances

We formally define the intersection of two TPOB-instances as follows. Let $\mathbf{I}_1 = (\pi_1, \nu_1)$ and $\mathbf{I}_2 = (\pi_2, \nu_2)$ be TPOB-instances over the same TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \mathrm{me}, \wp)$, and let $\otimes$ be a conjunction strategy. Then *intersection* of $\mathbf{I}_1$ and $\mathbf{I}_2$ under $\otimes$, denoted $\mathbf{I}_1 \cap_\otimes \mathbf{I}_2$, is the TPOB-instance (π, ν) over $\mathbf{S}$, where:

- $\pi(c) = \{o \in \pi_1(c) \cap \pi_2(c) \mid \nu_1(o) \cap_{\otimes} \nu_2(o) \text{ is defined}\}$, for all $c \in \mathcal{C}$.
- $\nu(o) = \nu_1(o) \cap_{\otimes} \nu_2(o)$, for all $o \in \pi(\mathcal{C})$.

The following example illustrates the intersection operation on two TPOB-instances.

Example 6.5 Let $\mathbf{S} = (\mathcal{C}, \sigma, \mathrm{me}, \wp)$ be the TPOB-schema of Example 4.2. Let $\mathbf{I}_1$ and $\mathbf{I}_2$ be the TPOB instances over $\mathbf{S}$ given in Example 4.3 and produced in Example 5.10, respectively. Then, the intersection of $\mathbf{I}_1$ and $\mathbf{I}_2$ under the conjunction strategy for ignorance is the TPOB-instance $\mathbf{I}_1 \cap_{\otimes_{ig}} \mathbf{I}_2 = (\pi, \nu)$ over $\mathbf{S}$, where π is given by $\pi(\mathsf{Priority}) = \{o_3\}$ and $\pi(c) = \emptyset$ for all other $c \in \mathcal{C}$, and ν is shown in Table 18. □

Table 18: ν resulting from intersection

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

6.3.2 Union of Values

We now define the union of two values. Let v_1 and v_2 be two values of the same classical or probabilistic type τ. The *union* of v_1 and v_2 under a disjunction strategy $\oplus$, denoted $v_1 \cup_{\oplus} v_2$, is inductively defined by:

- If τ is a classical type and $v_1 = v_2$, then $v_1 \cup_{\oplus} v_2 = v_1$.
- If τ is an atomic probabilistic type, then

$$\begin{aligned} v_1 \cup_{\oplus} v_2 \;=\; & \{(v, I_1) \in v_1 \mid v \in V_1 - V_2\} \cup \{(v, I_2) \in v_2 \mid v \in V_2 - V_1\} \cup \\ & \{(v, I_1 \oplus I_2) \mid (v, I_1) \in v_1,\ (v, I_2) \in v_2\}, \end{aligned}$$

  where $V_1 = \{v \mid (v, I) \in v_1\}$ and $V_2 = \{v \mid (v, I) \in v_2\}$.
- If τ is a probabilistic tuple type over the set of top-level attributes $\mathbf{A}$ and all $v_1.A \cup_{\oplus} v_2.A$ are defined, then $(v_1 \cup_{\oplus} v_2).A = v_1.A \cup_{\oplus} v_2.A$ for all $A \in \mathbf{A}$.

Otherwise, $v_1 \cup_{\oplus} v_2$ is undefined. The following example illustrates the above concept.

Example 6.6 Let time be the standard calendar w.r.t. $hour \sqsupseteq minute$, and let $\oplus$ be a disjunction strategy. Consider the values $v_1 = v_2 = (12, 30)$ and $v_3 = (12, 40)$ of the classical type time. Then, $v_1 \cup_{\oplus} v_2 = v_1$, while $v_1 \cup_{\oplus} v_3$ is undefined. Consider next the following values of the probabilistic type $[\![\mathsf{time}]\!]$:

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

Then, we have $v_1 \cup_{\oplus_{in}} v_2 = \{((9,00),[.3,.5]),((10,00),[.3,.6]),((11,00),[.2,.5]),\ ((12,00),[.2,.4])\}$, while $v_1 \cup_{\oplus_{in}} v_3 = \{((9,00),[.51,.7]),((10,00),[.3,.6]),((11,00),[.44,.8]),((12,00),[.2,.4])\}$. □

6.3.3 Union of TPOB-Instances

We are now ready to define the union of two TPOB-instances. Let $\mathbf{I}_1 = (\pi_1, \nu_1)$ and $\mathbf{I}_2 = (\pi_2, \nu_2)$ be two TPOB-instances over the same TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \mathrm{me}, \wp)$, and let $\oplus$ be a disjunction strategy. The *union* of $\mathbf{I}_1$ and $\mathbf{I}_2$ under $\oplus$, denoted $\mathbf{I}_1 \cup_\oplus \mathbf{I}_2$, is the TPOB-instance (π, ν) over $\mathbf{S}$, where:

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

The following example illustrates the union operation on two TPOB-instances.

Example 6.7 Let $\mathbf{S}$ be the TPOB-schema of Example 4.2. Let $\mathbf{I}_1$ and $\mathbf{I}_2$ be the TPOB-instances over $\mathbf{S}$ given in Example 4.3 and produced in Example 5.10, respectively. Then, the union of $\mathbf{I}_1$ and $\mathbf{I}_2$ under the disjunction strategy for positive correlation is the TPOB-instance $\mathbf{I}_1 \cup_{\oplus_{pc}} \mathbf{I}_2 = \mathbf{I}_1$. □

6.3.4 Difference of Values

We now define the difference of values. Let v_1 and v_2 be values of the same classical or probabilistic type τ. The *difference* of v_1 and v_2 under a difference strategy $\ominus$, denoted $v_1 -_\ominus v_2$, is inductively defined by:

- If τ is a classical type and $v_1 = v_2$, then $v_1 -_\ominus v_2 = v_1$.
- If τ is an atomic probabilistic type, then

$$v_1 -_\ominus v_2 \quad = \quad \{(v, I_1) \in v_1 \mid v \in V_1 - V_2\} \cup \{(v, I_1 \ominus I_2) \mid (v, I_1) \in v_1,\ (v, I_2) \in v_2\}\,,$$

  where $V_1 = \{v \mid (v, I) \in v_1\}$ and $V_2 = \{v \mid (v, I) \in v_2\}$.
- If τ is a probabilistic tuple type over the set of top-level attributes $\mathbf{A}$ and all $v_1.A -_\ominus v_2.A$ are defined, then $(v_1 -_\ominus v_2).A = v_1.A -_\ominus v_2.A$ for all $A \in \mathbf{A}$.

Otherwise, $v_1 -_\ominus v_2$ is undefined.

6.3.5 Difference of TPOB-Instances

We now define the difference of two TPOB-instances. Let $\mathbf{I}_1 = (\pi_1, \nu_1)$ and $\mathbf{I}_2 = (\pi_2, \nu_2)$ be TPOB-instances over the same TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$, and let $\ominus$ be a difference strategy. The *difference* of $\mathbf{I}_1$ and $\mathbf{I}_2$ under $\ominus$, denoted $\mathbf{I}_1 -_\ominus \mathbf{I}_2$, is the TPOB-instance (π, ν) over $\mathbf{S}$, where:

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

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, restricted selection, intersection, union, and difference trivially preserve consistency of schemas, as the input schemas coincide with the output schemas. Renaming and projection also preserve consistency of schemas, as they only modify type assignments. The following result shows that extraction and natural join, and thus also Cartesian product and conditional join, 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.*

We next concentrate on the preservation of coherence. Recall that the coherence of a TPOB-instance $\mathbf{I} = (\pi, \nu)$ over a TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \mathrm{me}, \wp)$ depends on π, $\mathcal{C}$, me, and $\wp$. The operations of selection, restricted selection, intersection, union, and difference 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, renaming and projection preserve coherence of instances, as they may only modify type and value assignments to classes and objects, respectively. The result below shows that natural join, and thus also Cartesian product and conditional join, preserve coherence of instances. It also shows that the 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* $\{c \in \mathcal{C} \mid c \text{ is characteristic for } \mathrm{ext}(d)(o) \text{ for some } d \in \mathbf{C} \text{ and some } o \in \pi(\mathbf{C})\} \subseteq \mathbf{C}$. *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 7 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 may (if we reason at a minute by minute level) requires 60 time points to be specified (Table 7 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, suppose we 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.

8.1 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.

8.2 Probability Distribution Functions

We now recall the well-known concept of a probability distribution function. In the sequel, we assume that S is a nonempty set of pairwise mutually exclusive events $s_1, \ldots, s_n$. A *probability distribution function* (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 distribution function 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 $s \in S$. The following are some other standard distribution functions (where we additionally assume that S has the linear order $s_1 < \ldots < s_n$):

- The *geometric distribution* over S for $0 < p < 1$, denoted $G_{S,p}$, is the function $G_{S,p}\colon S \to [0, 1]$ defined by $G_{S,p}(s_i) = p \cdot (1 - p)^i$ for all $s_i \in S$.
- The *binomial distribution* over S for $0 < p < 1$, denoted $B_{S,p}$, is the function $B_{S,p}\colon S \to [0, 1]$ defined by $B_{S,p}(s_i) = \binom{n}{i} \cdot p^i \cdot (1 - p)^{n-i}$ for all $s_i \in S$.

- The *Poisson distribution* over S for $\lambda > 0$, denoted $P_{S,\lambda}$, is the function $P_{S,\lambda}\colon S \to [0,1]$ defined by $P_{S,\lambda}(s_i) = e^{-\lambda} \cdot \lambda^i \,/\, i\,!$ for all $s_i \in S$.

8.3 Implicit Values of Probabilistic Types

An implicit value of a probabilistic type is a finite set of implicit tuples, a concept borrowed from [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 reader may wonder why both constraints C, D are needed. The reason is that 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 (by performing an AND) rather than computing a new distribution (which would be needed if D were not used). 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 $[A_1\colon v_1, \ldots, A_k\colon v_k]$, 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 3.3 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 $\varepsilon(v) = \{(v', [l \cdot \delta(v'), u \cdot \delta(v')]) \mid \exists (C, D, l, u, \delta) \in v\colon v' \in \mathrm{sol}(C)\}$. ε is a *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 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.

8.4 Implicit TPOB-Instances

Implicit TPOB-instances associate an implicit value of appropriate probabilistic type 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) $\pi\colon \mathcal{C} \to 2^{\mathcal{O}}$ 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 6 and the value assignment ν in Table 19. □

Table 19: 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 the algebraic operations of selection, restricted selection, natural join, and the set operations of intersection and union on implicit TPOB-instances. The operations of projection, extraction, Cartesian product, and conditional 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 and every operation of the implicit TPOB-algebra correctly implements its explicit counterpart. Figure 1 provides a diagrammatic representation of what these results look like for unary operators (a similar figure can be shown for binary operators). For unary algebraic operators *op*, the correctness theorems are of the form $\varepsilon(op^i(\mathbf{I})) = op^e(\varepsilon(\mathbf{I}))$. We use op^i (resp. op^e) to denote 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.

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 $v = [A_1 \colon v_1, \ldots, A_k \colon v_k]$ 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 $v = \{(C_1, D_1, l_1, u_1, \delta_1), \ldots, (C_k, D_k, l_k, u_k, \delta_k)\}$ 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 atomic selection condition 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 atomic selection condition $\phi = \mathsf{Delivery} > (18, 25)$ 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 $\mathsf{Delivery}$. □

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,*

$$\sigma_\phi(\varepsilon(\mathbf{I})) \quad = \quad \varepsilon(\sigma_\phi(\mathbf{I}))\,. \tag{1}$$

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,*

$$\sigma^r_\phi(\varepsilon(\mathbf{I})) \quad = \quad \varepsilon(\sigma^r_\phi(\mathbf{I}))\,. \tag{2}$$

9.3 Renaming

To define renaming on implicit TPOB-instances, we need to define renaming on implicit values, which is then extended to implicit TPOB-instances in the same way as in Section 5.3.

Let C be a constraint for the classical type τ, and let N be a renaming condition for τ. The *renaming* on C with respect to N, denoted $\delta_N(C)$, is obtained from C by replacing every value v_i in C by $\delta_N(v_i)$.

Let N be a renaming condition of the form $P \leftarrow P'$ for the probabilistic tuple type $\tau = [A_1\colon \tau_1, \ldots, A_n\colon \tau_n]$. Let $v = [A_1\colon v_1, \ldots, A_n\colon v_n]$ be an implicit value of τ. The *renaming* on v with respect to N, denoted $\delta_N(v)$, is defined by:

- If $P = A_i$ and $P' = A_i{}'$, then $\delta_N(v)$ is obtained from v by replacing A_i by $A_i{}'$.

- If $P = A_i.[\![R]\!]$, $P' = A_i.[\![R']\!]$, and v_i is a value of an atomic probabilistic type, then $\delta_N(v)$ is obtained from v by replacing every $(C, D, l, u, \rho) \in v_i$ by $(\delta_{R \leftarrow R'}(C), \delta_{R \leftarrow R'}(D), l, u, \rho \circ \delta^{-1}_{R \leftarrow R'})$, where $\delta^{-1}_{R \leftarrow R'}$ denotes the inverse to $\delta_{R \leftarrow R'} \colon \mathrm{sol}(D) \to \mathrm{sol}(\delta_{R \leftarrow R'}(D))$.
- If $P = A_i.R$, $P' = A_i.R'$, and v_i is not a value of an atomic probabilistic type, then $\delta_N(v)$ is obtained from v by replacing v_i by $\delta_{R \leftarrow R'}(v_i)$.

Let $N = P_1, \ldots, P_l \leftarrow P_1', \ldots, P_l'$ be a renaming condition for τ. The *renaming* on v with respect to N, denoted $\delta_N(v)$, is defined as the simultaneous renaming on v with respect to all $P_i \leftarrow P_i'$.

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

Theorem 9.5 (correctness of renaming) *Let* $\mathbf{I}$ *be an implicit TPOB-instance over the TPOB-schema* $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$*, and let* N *be a renaming condition for every* $\sigma(c)$ *with* $c \in \mathcal{C}$*. Then,*

$$\delta_N(\varepsilon(\mathbf{I})) \quad = \quad \varepsilon(\delta_N(\mathbf{I}))\,.$$

9.4 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:

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

- 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 9.6 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 9.7 (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,*

$$\varepsilon(\mathbf{I}_1) \bowtie_\otimes \varepsilon(\mathbf{I}_2) \quad = \quad \varepsilon(\mathbf{I}_1 \bowtie_\otimes \mathbf{I}_2)\,. \tag{3}$$

9.5 Intersection, Union, and Difference

To define the intersection, union, and difference of two TPOB-instances, we need to define the intersection, union, and difference, respectively, of two implicit values, which are then extended to implicit TPOB-instances using methods similar to those of Section 6.3. Intersection of two implicit values was defined in the preceding section. Union and difference of two implicit values are defined below.

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 $\oplus$ (resp., $\ominus$) be a disjunction (resp., difference) strategy. The *union* of v_1 and v_2 under $\oplus$, denoted $v_1 \cup_\oplus v_2$, is inductively defined as follows:

- If τ is a classical type and $v_1 = v_2$, then $v_1 \cup_\oplus v_2 = v_1$.
- If τ is an atomic probabilistic type, then

$$\begin{aligned} v_1 \cup_\oplus v_2 \;=\; & \{(C_1 \wedge \neg\widehat{C}_2, D_1, l_1, u_1, \delta_1) \mid (C_1, D_1, l_1, u_1, \delta_1) \in v_1,\ \mathrm{sol}(C_1 \wedge \neg\widehat{C}_2) \neq \emptyset\} \cup \\ & \{(C_2 \wedge \neg\widehat{C}_1, D_2, l_2, u_2, \delta_2) \mid (C_2, D_2, l_2, u_2, \delta_2) \in v_2,\ \mathrm{sol}(C_2 \wedge \neg\widehat{C}_1) \neq \emptyset\} \cup \\ & \{((\#), (v), l, u, U) \mid \exists (C_1, D_1, l_1, u_1, \delta_1) \in v_1, (C_2, D_2, l_2, u_2, \delta_2) \in v_2 : \\ & \qquad v \in \mathrm{sol}(C_1 \wedge C_2),\ [l, u] = [l_1 \cdot \delta_1(v), u_1 \cdot \delta_1(v)] \oplus [l_2 \cdot \delta_2(v), u_2 \cdot \delta_2(v)]\}\,, \end{aligned}$$

  where $\widehat{C}_i$, $i \in \{1, 2\}$, denotes the logical disjunction of all C_i such that $(C_i, D_i, l_i, u_i, \delta_i) \in v_i$.
- If τ is a probabilistic tuple type over the set of top-level attributes $\mathbf{A}$ and all $v_1.A \cup_\oplus v_2.A$ are defined, then $(v_1 \cup_\oplus v_2).A = v_1.A \cup_\oplus v_2.A$ for all $A \in \mathbf{A}$.
- Otherwise, $v_1 \cup_\oplus v_2$ is undefined.

The *difference* of v_1 and v_2 under $\ominus$, denoted $v_1 -_\ominus v_2$, is inductively defined as follows:

- If τ is a classical type and $v_1 = v_2$, then $v_1 -_\ominus v_2 = v_1$.
- If τ is an atomic probabilistic type, then

$$\begin{aligned} v_1 -_\ominus v_2 \;=\; & \{(C_1 \wedge \neg\widehat{C}_2, D_1, l_1, u_1, \delta_1) \mid (C_1, D_1, l_1, u_1, \delta_1) \in v_1,\ \mathrm{sol}(C_1 \wedge \neg\widehat{C}_2) \neq \emptyset\} \cup \\ & \{((\#), (v), l, u, U) \mid \exists (C_1, D_1, l_1, u_1, \delta_1) \in v_1, (C_2, D_2, l_2, u_2, \delta_2) \in v_2 : \\ & \qquad v \in \mathrm{sol}(C_1 \wedge C_2),\ [l, u] = [l_1 \cdot \delta_1(v), u_1 \cdot \delta_1(v)] \ominus [l_2 \cdot \delta_2(v), u_2 \cdot \delta_2(v)]\}\,, \end{aligned}$$

  where $\widehat{C}_i$, $i \in \{1, 2\}$, denotes the logical disjunction of all C_i such that $(C_i, D_i, l_i, u_i, \delta_i) \in v_i$.
- If τ is a probabilistic tuple type over the set of top-level attributes $\mathbf{A}$ and all $v_1.A -_\ominus v_2.A$ are defined, then $(v_1 -_\ominus v_2).A = v_1.A -_\ominus v_2.A$ for all $A \in \mathbf{A}$.
- Otherwise, $v_1 -_\ominus v_2$ is undefined.

The following theorem shows that the intersection, union, and difference of implicit TPOB-instances correctly implement their counterparts on explicit TPOB-instances. That is, the mapping ε commutes with $\cap_\otimes$, $\cup_\oplus$, and $-_\ominus$, respectively.

Theorem 9.8 (correctness of intersection, union, and difference) *Let $\mathbf{I}_1$ and $\mathbf{I}_2$ be two implicit TPOB-instances over the same TPOB-schema $\mathbf{S}$, and let $\otimes$ (resp., $\oplus$, $\ominus$) be a conjunction (resp., disjunction, difference) strategy. Then,*

$$\varepsilon(\mathbf{I}_1) \cap_\otimes \varepsilon(\mathbf{I}_2) \;=\; \varepsilon(\mathbf{I}_1 \cap_\otimes \mathbf{I}_2)\,, \tag{4}$$

$$\varepsilon(\mathbf{I}_1) \cup_\oplus \varepsilon(\mathbf{I}_2) \;=\; \varepsilon(\mathbf{I}_1 \cup_\oplus \mathbf{I}_2)\,, \tag{5}$$

$$\varepsilon(\mathbf{I}_1) -_\ominus \varepsilon(\mathbf{I}_2) \;=\; \varepsilon(\mathbf{I}_1 -_\ominus \mathbf{I}_2)\,. \tag{6}$$

9.6 Compression Functions

The implicit operations of natural join, intersection, union, and difference may generate implicit TPOB-instances that contain a large number of implicit tuples. Adopting an idea from [13], we now define compression functions through which such implicit TPOB-instances can be made more compact.

A *compression function* Γ for an atomic probabilistic type τ is a function that maps every implicit value v of τ to an implicit value $\Gamma(v)$ of τ such that (i) $|\Gamma(v)| \leq |v|$, and (ii) there exists a bijection between $\varepsilon(v)$ and $\varepsilon(\Gamma(v))$ that maps each $(v, [l, u]) \in \varepsilon(v)$ to a pair $(v, [l, u']) \in \varepsilon(\Gamma(v))$ such that $l \leq u' \leq u$.

Example 9.9 Let τ be an atomic probabilistic type. The *same-distribution compression function* Γ maps every implicit value v of τ to the implicit value $\Gamma(v)$, which is obtained from v by iteratively replacing any two distinct $(C_1, D_1, l, u, \delta), (C_2, D_2, l, u, \delta) \in v$ with $\mathrm{sol}(D_1) = \mathrm{sol}(D_2)$ by $(C_1 \vee C_2, D_1, l, u, \delta)$.

We now define the compression of implicit values of probabilistic types. Here, we assume that for every atomic probabilistic type τ, we have some compression function Γ_τ. More formally, let v be either a value of a classical type τ, or an implicit value of a probabilistic type τ. The *compression* of v, denoted $\Gamma(v)$, is inductively defined as follows:

- If τ is a classical type, then $\Gamma(v) = v$.
- If τ is an atomic probabilistic type, then $\Gamma(v) = \Gamma_\tau(v)$.
- If τ is a probabilistic tuple type over the set of top-level attributes $\mathbf{A}$, then $\Gamma(v).A = \Gamma(v.A)$ for all $A \in \mathbf{A}$.

We finally define the compression of implicit TPOB-instances as follows. Let $\mathbf{I} = (\pi, \nu)$ be a TPOB-instance over the TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$. The *compression* of $\mathbf{I}$, denoted $\Gamma(\mathbf{I})$, is defined as the TPOB-instance (π, ν') over $\mathbf{S}$, where $\nu'(o) = \Gamma(\nu(o))$ for all $o \in \pi(\mathcal{C})$.

10 Conclusions

There are numerous applications where object models are naturally used and where temporal uncertainty is a critical part of the application. A natural place to start is shipping and transportation companies. These companies utilize numerous different kinds of resources (airplanes, ships, and trucks, all with different capacities, fuel requirements, fuel consumption, servicing needs, and other properties) and ship hundreds of thousands of items daily. The properties of these vehicles vary dramatically (as the properties of ships, planes, and trucks are all very different). 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. The data model allows to associate with events e a set of possible time points T and with each time point $t \in T$ an interval for the probability that e occurred at t. We have presented explicit object base instances, where the sets of time points along with their probability intervals are simply 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. 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. This work was partially supported by the Army Research Lab under contract number DAAL0197K0135, the Army Research Office under grant number DAAD190010484, by DARPA/RL contract number F306029910552, by the ARL CTA on Advanced Decision Architectures, by NSF grant 937756, by the Austrian Science Fund Project N Z29-INF, by a DFG grant, and by a Marie Curie Individual Fellowship of the European Community (Disclaimer: The authors are solely responsible for information communicated and the European Commission is not responsible for any views or results expressed).

Appendix A: Proofs for Section 7

Proof of Theorem 7.1. (a) Let $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$, and let $\Xi_{\mathbf{C}}(\mathbf{S}) = \mathbf{S}' = (\mathcal{C}', \sigma', \Rightarrow', \mathrm{me}', \wp')$. Assume that $\mathbf{S}$ is consistent. That is, there exists a model $\varepsilon\colon \mathcal{C} \to 2^{\mathcal{O}}$ of $\mathbf{S}$. Let the mapping $\varepsilon'\colon \mathcal{C}' \to 2^{\mathcal{O}}$ be defined by $\varepsilon'(c) = \varepsilon(c)$ for all $c \in \mathcal{C}'$. We now show that ε' is a model of $\mathbf{S}'$. C1 holds, as $\varepsilon(c) \neq \emptyset$ for all classes $c \in \mathcal{C}$ implies $\varepsilon'(c) \neq \emptyset$ for all classes $c \in \mathcal{C}'$. We next show C2. Consider two classes $c_1, c_2 \in \mathcal{C}'$ such that $c_1 \Rightarrow' c_2$. That is, some path $d_1 \Rightarrow d_2 \Rightarrow \cdots \Rightarrow d_k$ exists such that $d_1 = c_1$, $d_k = c_2$, and $d_2, \ldots, d_{k-1} \in \mathcal{C} - \mathcal{C}'$. As $\varepsilon(d_1) \subseteq \varepsilon(d_2) \subseteq \cdots \subseteq \varepsilon(d_k)$, it thus follows $\varepsilon(c_1) \subseteq \varepsilon(c_2)$. We now prove that C3 holds. Let $c_1, c_2 \in \mathcal{C}'$ be two distinct classes that belong to the same cluster $\mathcal{P}' \in \mathrm{me}'(c)$ for some $c \in \mathcal{C}'$. That is, there exists a cluster $\mathcal{P} \in \mathrm{me}(c)$ such that, for $i \in \{1, 2\}$, either c_i belongs to $\mathcal{P}$ or c_i is a proper subclass of a class in $\mathcal{P}$. As C2 and C3 hold for ε, it thus follows that $\varepsilon(c_1) \cap \varepsilon(c_2) = \emptyset$. This shows that C3 holds. We finally prove C4. Consider two classes $c_1, c_2 \in \mathcal{C}'$ such that $c_1 \Rightarrow' c_2$. That is, some path $d_1 \Rightarrow d_2 \Rightarrow \cdots \Rightarrow d_k$ exists such that $d_1 = c_1$, $d_k = c_2$, and $d_2, \ldots, d_{k-1} \in \mathcal{C} - \mathcal{C}'$. Moreover, it holds $\wp'(c_1, c_2) = \prod_{i=1}^{k-1} \wp(d_i, d_{i+1})$. As C4 holds for ε, it follows that $|\varepsilon(d_i)| = \wp(d_i, d_{i+1}) \cdot |\varepsilon(d_{i+1})|$ for all $i \in \{1, \ldots, k-1\}$. This shows that $|\varepsilon(c_1)| = \prod_{i=1}^{k-1} \wp(d_i, d_{i+1}) \cdot |\varepsilon(c_2)|$, that is, $|\varepsilon'(c_1)| = \wp'(c_1, c_2) \cdot |\varepsilon'(c_2)|$. This proves C4.

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

$$\varepsilon(c) \;=\; \varepsilon_1(c_1) \times \varepsilon_2(c_2), \;\; \text{for all } c = (c_1, c_2) \in \mathcal{C}\,.$$

We now show that ε is a model of $\mathbf{S}$. We first prove C1. Since $\varepsilon_1(c_1) \neq \emptyset$ for all classes $c_1 \in \mathcal{C}_1$ and $\varepsilon_2(c_2) \neq \emptyset$ for all classes $c_2 \in \mathcal{C}_2$, we get $\varepsilon(c) \neq \emptyset$ for all classes $c \in \mathcal{C}$. We next show C2 and C4. Let $c = (c_1, c_2), d = (d_1, d_2) \in \mathcal{C}$ with $c \Rightarrow d$. Without loss of generality, we can assume that $c_1 \Rightarrow_1 d_1$ and $c_2 = d_2$. Since ε_1 is a model of $\mathbf{S}_1$, it holds that $\varepsilon_1(c_1) \subseteq \varepsilon_1(d_1)$ and $|\varepsilon_1(c_1)| = \wp_1(c_1, d_1) \cdot |\varepsilon_1(d_1)|$. Hence, it immediately follows $\varepsilon(c) \subseteq \varepsilon(d)$ and $|\varepsilon(c)| = \wp(c, d) \cdot |\varepsilon(d)|$. We finally prove C3. Let $c, d \in \mathcal{C}$ be two distinct classes that belong to the same cluster $\mathcal{P} \in \bigcup \mathrm{me}(\mathcal{C})$. Without loss of generality, we can assume

that $c_1, d_1 \in \mathcal{C}_1$ belong to the same cluster $\mathcal{P}_1 \in \bigcup \mathrm{me}_1(\mathcal{C}_1)$ and that $c_2 = d_2$. Since ε_1 is a model of $\mathbf{S}_1$, it holds that $\varepsilon_1(c_1) \cap \varepsilon_1(d_1) = \emptyset$. Thus, $\varepsilon(c) \cap \varepsilon(d) = \emptyset$. □

Proof of Theorem 7.2. (a) Let $\mathbf{I} = (\pi, \nu)$ and $\Xi_{\mathbf{C}}(\mathbf{I}) = \mathbf{I}' = (\pi', \nu')$. Let $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ and $\Xi_{\mathbf{C}}(\mathbf{S}) = \mathbf{S}' = (\mathcal{C}', \sigma', \Rightarrow', \mathrm{me}', \wp')$. Towards a contradiction, suppose that $\mathbf{I}'$ is not coherent. That is, there exists a class $c \in \mathcal{C}'$ and an object $o \in \pi'(c)$ such that $p_1, p_2 \in \mathrm{ext}'(c)(o)$ with $p_1 \neq p_2$. Hence, there are at least two distinct classes $d_1, d_2 \in \mathcal{C}'$ such that (i) $o \in (\pi')^\star(d_i)$ and $c\,(\Rightarrow')^\star\, d_i$, and (ii) d_i is minimal under $(\Rightarrow')^\star$ with (i), and (iii) p_i is the product of edge probabilities from c up to d_i. As $\mathbf{C}$ contains all characteristic classes for $\mathrm{ext}(c)(o)$, there are at least two distinct classes $d_i \in \mathcal{C}$ such that (i) $o \in \pi^\star(d_i)$ and $c \Rightarrow^\star d_i$, and (ii) d_i is minimal under $\Rightarrow^\star$ with (i). This implies that $p_1, p_2 \in \mathrm{ext}(c)(o)$. But this contradicts $\mathbf{S}$ being coherent.

(b) Let $\mathbf{S}_1 = (\mathcal{C}_1, \sigma_1, \Rightarrow_1, \mathrm{me}_1, \wp_1)$, $\mathbf{S}_2 = (\mathcal{C}_2, \sigma_2, \Rightarrow_2, \mathrm{me}_2, \wp_2)$, and $\mathbf{S}_1 \bowtie \mathbf{S}_2 = \mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$. Let $\mathbf{I}_1 = (\pi_1, \nu_1)$, $\mathbf{I}_2 = (\pi_2, \nu_2)$, and $\mathbf{I}_1 \bowtie \mathbf{I}_2 = \mathbf{I} = (\pi, \nu)$. Towards a contradiction, suppose that $\mathbf{I}$ is not coherent. That is, there exists a class $c = (c_1, c_2) \in \mathcal{C}$ and an object $o = (o_1, o_2) \in \pi(c)$ such that $p^1, p^2 \in \mathrm{ext}(c)(o)$ with $p^1 \neq p^2$. Hence, there are classes $d^1 = (d_1^1, d_2^1), d^2 = (d_1^2, d_2^2) \in \mathcal{C}$ such that (i) $o \in \pi^\star(d^i)$ and $c \Rightarrow^\star d^i$, (ii) d^i is minimal under $\Rightarrow^\star$ with (i), and (iii) p^i is the product of edge probabilities from c up to d^i. Thus, c, d^1, and d^2 are pairwise distinct. Moreover, $p^1 = p_1^1 \cdot p_2^1$ and $p^2 = p_1^2 \cdot p_2^2$, where p_j^i is the product of edge probabilities from c_j up to d_j^i. Suppose now $c_1 = d_1^1$. Then, $o \in \pi^\star((c_1, d_2^2))$, $c \Rightarrow^\star (c_1, d_2^2)$, and $(c_1, d_2^2) \Rightarrow^\star d^2$. By the minimality of d^2, it then follows $c_1 = d_1^1 = d_1^2$, and thus $p_1^1 = p_1^2 = 1$. Moreover, if $d_1^1 = d_1^2$, then $p_1^1 = p_1^2$. Thus, as $p_1 \neq p_2$, we can assume without loss of generality that c_1, d_1^1, and d_1^2 are pairwise distinct. As $\mathbf{S}_1$ is coherent, there is some $d_1^0 \in \mathcal{C}_1$ such that $o_1 \in \pi^\star(d_1^0)$, $c_1 \Rightarrow^\star d_1^0$, $d_1^0 \Rightarrow^\star d_1^1$, and $d_1^0 \Rightarrow^\star d_1^2$. Without loss of generality, we can assume that $d_1^0 \neq d_1^1$. Hence, $o \in \pi^\star((d_1^0, d_2^1))$, $c \Rightarrow^\star (d_1^0, d_2^1)$, $(d_1^0, d_2^1) \Rightarrow^\star d^1$, and $(d_1^0, d_2^1) \neq d^1$. But this contradicts d^1 being minimal under $\Rightarrow^\star$ with (i). Hence, $\mathbf{I}$ is coherent. □

Appendix B: Proofs for Section 9

For the proof of Theorem 9.2, we need the following lemma, which says that the valuation of path expressions under implicit values correctly implements the valuation of path expressions under explicit values. Here, the mapping ε is extended to generalized implicit values as follows: Every generalized implicit value $w = \{(C_1, D_1, l_1, u_1, \delta_1, S), \ldots, (C_k, D_k, l_k, u_k, \delta_k, S)\}$ is associated with the generalized explicit value $\varepsilon(w) = \{(w', I, S) \mid (w', I) \in \varepsilon(\{(C_1, D_1, l_1, u_1, \delta_1), \ldots, (C_k, D_k, l_k, u_k, \delta_k)\})\}$.

Lemma B.1 *Let P be a path expression for the probabilistic type τ, and v be an implicit value of τ. Then,*

$$\varepsilon(v.P) \;=\; \varepsilon(v).P\,. \tag{7}$$

Proof. It is sufficient to show that $\varepsilon(v.P) = \varepsilon(v).P$ holds for every implicit value v of an atomic probabilistic type τ. Let $v = \{(C_1, D_1, l_1, u_1, \delta_1), \ldots, (C_k, D_k, l_k, u_k, \delta_k)\}$ and $P = [\![R]\!]$. Then,

$$\begin{aligned}
\varepsilon(v.P) &= \varepsilon(\{(C_1, D_1, l_1, u_1, \delta_1, R), \ldots, (C_k, D_k, l_k, u_k, \delta_k, R)\}) \\
&= \{(w', I, R) \mid (w', I) \in \varepsilon(\{(C_1, D_1, l_1, u_1, \delta_1), \ldots, (C_k, D_k, l_k, u_k, \delta_k)\})\} \\
&= \{(w', I, R) \mid (w', I) \in \varepsilon(v)\} \\
&= \varepsilon(v).P\,. \;\square
\end{aligned}$$

Proof of Theorem 9.2. It is sufficient to show that the valuation of atomic selection conditions with respect to implicit TPOB-instances is correct. Let $\mathbf{I} = (\pi, \nu)$ be an implicit TPOB-instance over the TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$, and let $o \in \pi(\mathcal{C})$. Let $\oplus$ be the disjunction strategy for mutual exclusion. Then,

- $\mathrm{prob}_{\mathbf{I},o}(in(c)) = [\min(\mathrm{ext(c)}(o)), \max(\mathrm{ext(c)}(o))] = \mathrm{prob}_{\varepsilon(\mathbf{I}),o}(in(c))$.
- 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. That is, the intervals $[l', u']$ such that $(w, [l', u'], S) \in \varepsilon(V)$ and $w.S\,\theta\,v$, if V is defined. By Lemma B.1 and as $\varepsilon(\{((\#), (\nu(o)), 1, 1, U, P)\}) = \{(\nu(o), [1, 1], P)\}$, it follows that

  $$\mathrm{prob}_{\mathbf{I},o}(P\,\theta\,v) = \mathrm{prob}_{\varepsilon(\mathbf{I}),o}(P\,\theta\,v)\,.$$

- 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. That is, the list of all intervals $[{l_1}', {u_1}'] \otimes [{l_2}', {u_2}']$ such that $(v_i, [{l_i}', {u_i}'], S_i) \in \varepsilon(V_i)$ and $v_1.S_1\,\theta\,v_2.S_2$, if V_1 and V_2 are defined. By Lemma B.1 and as $\varepsilon(\{((\#), (\nu(o)), 1, 1, U, P_i)\}) = \{(\nu(o), [1, 1], P_i)\}$, it follows that

  $$\mathrm{prob}_{\mathbf{I},o}(P_1\,\theta_{\otimes}\,P_2) = \mathrm{prob}_{\varepsilon(\mathbf{I}),o}(P_1\,\theta_{\otimes}\,P_2)\,.$$

This shows that $\mathrm{prob}_{\mathbf{I},o}(\phi) = \mathrm{prob}_{\varepsilon(\mathbf{I}),o}(\phi)$ for all atomic selection conditions ϕ. Notice that this statement also includes that $\mathrm{prob}_{\mathbf{I},o}(\phi)$ is defined iff $\mathrm{prob}_{\varepsilon(\mathbf{I}),o}(\phi)$ is defined. □

Proof of Theorem 9.4. It is sufficient to show that the restricted selection on implicit values of atomic probabilistic types is correct. Let $\tau = [A_1\colon \tau_1, \ldots, A_k\colon \tau_k]$ be a probabilistic tuple type, and let $v = [A_1\colon v_1, \ldots, A_k\colon v_k]$ be an implicit value of τ. Let $\phi = A_i.C$, where $i \in \{1, \ldots, k\}$ and C is a constraint, and let τ_i be an atomic probabilistic type. Then,

$$\begin{aligned} & \varepsilon(\{(C \wedge C', D, l, u, \delta) \mid (C', D, l, u, \delta) \in v_i,\ \mathrm{sol}(C \wedge C') \neq \emptyset\}) \\ = \ & \{({v_i}', [l \cdot \delta({v_i}'), u \cdot \delta({v_i}')]) \mid \exists (C', D, l, u, \delta) \in v_i\colon {v_i}' \in \mathrm{sol}(C \wedge C')\} \\ = \ & \{({v_i}', [l', u']) \in \varepsilon(v_i) \mid {v_i}' \in \mathrm{sol}(C)\}\,. \end{aligned}$$

This shows that $\varepsilon(\sigma^r_{\phi}(v)) = \sigma^r_{\phi}(\varepsilon(v))$. □

Proof of Theorem 9.5. It is sufficient to show that the renaming of single attributes inside implicit values of atomic probabilistic types is correct. Let N be a renaming condition of the form $A_i.[\![R]\!] \leftarrow A_i.[\![R']\!]$

for the probabilistic tuple type $\tau = [A_1 \colon \tau_1, \ldots, A_n \colon \tau_n]$, where τ_i is an atomic probabilistic type, and let $v = [A_1 \colon v_1, \ldots, A_n \colon v_n]$ be an implicit value of τ. Then,

$$\begin{aligned}
& \varepsilon(\{(\delta_{R\leftarrow R'}(C), \delta_{R\leftarrow R'}(D), l, u, \rho \circ \delta^{-1}_{R\leftarrow R'}) \mid (C, D, l, u, \rho) \in v_i\}) \\
= \; & \{(v_i'', [l \cdot \rho(\delta^{-1}_{R\leftarrow R'}(v_i'')), u \cdot \rho(\delta^{-1}_{R\leftarrow R'}(v_i''))]) \mid \exists (C, D, l, u, \rho) \in v_i \colon v_i'' \in \mathrm{sol}(\delta_{R\leftarrow R'}(C))\} \\
= \; & \{(\delta_{R\leftarrow R'}(v_i'), [l \cdot \rho(v_i'), u \cdot \rho(v_i')]) \mid \exists (C, D, l, u, \rho) \in v_i \colon v_i' \in \mathrm{sol}(C)\} \\
= \; & \{(\delta_{R\leftarrow R'}(v_i'), [l', u']) \mid (v_i', [l', u']) \in \varepsilon(v_i)\}\,.
\end{aligned}$$

This shows that $\varepsilon(\delta_N(v)) = \delta_N(\varepsilon(v))$. □

Proof of Theorem 9.7. It is sufficient to show that the intersection of two implicit values of the same atomic probabilistic type is correct. Let v_1 and v_2 be two values of the same atomic probabilistic type, and let $\otimes$ be a conjunction strategy. Then,

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

This shows that $\varepsilon(v_1 \cap_\otimes v_2)$ is defined iff $\varepsilon(v_1) \cap_\otimes \varepsilon(v_2)$ is defined. Moreover, if they are both defined, then $\varepsilon(v_1 \cap_\otimes v_2) = \varepsilon(v_1) \cap_\otimes \varepsilon(v_2)$. □

Proof of Theorem 9.8. Equation (4) follows immediately from the proof of Theorem 9.7. We next prove Equation (5). It is sufficient to show that the union of two implicit values of the same atomic probabilistic type is correct. Let v_1 and v_2 be two values of the same atomic probabilistic type, and let $\oplus$ be a disjunction strategy. Then,

$$\begin{aligned}
& \varepsilon(v_1 \cup_\oplus v_2) \\
= \; & \varepsilon(\{(C_1 \wedge \neg\widehat{C}_2, D_1, l_1, u_1, \delta_1) \mid (C_1, D_1, l_1, u_1, \delta_1) \in v_1,\ \mathrm{sol}(C_1 \wedge \neg\widehat{C}_2) \neq \emptyset\}) \cup \\
& \varepsilon(\{(C_2 \wedge \neg\widehat{C}_1, D_2, l_2, u_2, \delta_2) \mid (C_2, D_2, l_2, u_2, \delta_2) \in v_2,\ \mathrm{sol}(C_2 \wedge \neg\widehat{C}_1) \neq \emptyset\}) \cup \\
& \varepsilon(\{((\#), (v), l, u, U) \mid \exists (C_1, D_1, l_1, u_1, \delta_1) \in v_1, (C_2, D_2, l_2, u_2, \delta_2) \in v_2 \colon \\
& \qquad v \in \mathrm{sol}(C_1 \wedge C_2),\ [l, u] = [l_1 \cdot \delta_1(v), u_1 \cdot \delta_1(v)] \oplus [l_2 \cdot \delta_2(v), u_2 \cdot \delta_2(v)]\}) \\
= \; & \{(v_1', [l_1 \cdot \delta_1(v_1'), u_1 \cdot \delta_1(v_1')]) \mid \exists (C_1, D_1, l_1, u_1, \delta_1) \in v_1 \colon v_1' \in \mathrm{sol}(C_1) - \mathrm{sol}(\widehat{C}_2)\} \cup \\
& \{(v_2', [l_2 \cdot \delta_2(v_2'), u_2 \cdot \delta_2(v_2')]) \mid \exists (C_2, D_2, l_2, u_2, \delta_2) \in v_2 \colon v_2' \in \mathrm{sol}(C_2) - \mathrm{sol}(\widehat{C}_1)\} \cup \\
& \{(v, [l_1', u_1'] \oplus [l_2', u_2']) \mid (v, [l_1', u_1']) \in \varepsilon(v_1), (v, [l_2', u_2']) \in \varepsilon(v_2)\} \\
= \; & \{(v_1', [l_1', u_1']) \in \varepsilon(v_1) \mid v_1' \notin \mathrm{sol}(\widehat{C}_2)\} \cup \{(v_2', [l_2', u_2']) \in \varepsilon(v_2) \mid v_2' \notin \mathrm{sol}(\widehat{C}_1)\} \cup \\
& \{(v, [l_1', u_1'] \oplus [l_2', u_2']) \mid (v, [l_1', u_1']) \in \varepsilon(v_1), (v, [l_2', u_2']) \in \varepsilon(v_2)\} \\
= \; & \varepsilon(v_1) \cup_\oplus \varepsilon(v_2)\,.
\end{aligned}$$

We finally prove Equation (6). Again, it is sufficient to show that the difference of two implicit values of the same atomic probabilistic type is correct. Let v_1 and v_2 be two values of the same atomic probabilistic type,

and let $\ominus$ be a difference strategy. Then,

$$
\begin{aligned}
& \varepsilon(v_1 -_{\ominus} v_2) \\
= \;& \varepsilon(\{(C_1 \wedge \neg\widehat{C}_2, D_1, l_1, u_1, \delta_1) \mid (C_1, D_1, l_1, u_1, \delta_1) \in v_1,\ \mathrm{sol}(C_1 \wedge \neg\widehat{C}_2) \neq \emptyset\}) \cup \\
& \varepsilon(\{((\#), (v), l, u, U) \mid \exists (C_1, D_1, l_1, u_1, \delta_1) \in v_1, (C_2, D_2, l_2, u_2, \delta_2) \in v_2 : \\
& \qquad v \in \mathrm{sol}(C_1 \wedge C_2),\ [l, u] = [l_1 \cdot \delta_1(v), u_1 \cdot \delta_1(v)] \ominus [l_2 \cdot \delta_2(v), u_2 \cdot \delta_2(v)]\}) \\
= \;& \{(v_1', [l_1', u_1']) \in \varepsilon(v_1) \mid v_1' \notin \mathrm{sol}(\widehat{C}_2)\} \cup \\
& \{(v, [l_1', u_1'] \ominus [l_2', u_2']) \mid (v, [l_1', u_1']) \in \varepsilon(v_1), (v, [l_2', u_2']) \in \varepsilon(v_2)\} \\
= \;& \varepsilon(v_1) -_{\ominus} \varepsilon(v_2)\,. \;\square
\end{aligned}
$$

Appendix C: Table of Commonly Used Symbols

Table 20: Notation

Symbol	Description	Section
$[\![\tau]\!]$	atomic probabilistic type	3.3
$\mathcal{C}$	set of classes	4.1
$\sigma(c)$	type assignment of a class c	4.1
$\Rightarrow$	immediate subclass relationship	4.1
$\Rightarrow^\star$	subclass relationship (i.e., reflexive and transitive closure of $\Rightarrow$)	4.1
$\mathrm{me}(c)$	partition of the subclasses of a class c	4.1
$\wp(c_1, c_2)$	conditional probability that an object belongs to a class c_1 given that it belongs to an immediate superclass c_2	4.1
$\mathbf{S} = (\mathcal{C}, \sigma, \mathrm{me}, \wp)$	TPOB-schema	4.1
ζ	mapping from the set of classes to a set of objects	4.1
$\pi(c)$	set of object identifiers in a class c	4.3
$\nu(o)$	value of an object $o \in \pi(c)$	4.3
$\mathbf{I} = (\pi, \nu)$	TPOB-instance	4.3
$\mathrm{ext}(c)(o)$	probabilities with which an object o belongs to a class c	4.3
$\pi^\star(c)$	object identifiers in c and all subclasses of 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.* To appear in *Annals of Mathematics and Artificial Intelligence*. 2002.

[3] E. I. Altman. *Financial Ratios, Discriminant Analysis, and the Prediction of a Corporate Bankruptcy*. Journal of Finance. pp. 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*, pages 40–57. Elsevier Science Publishers, 1989.

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

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

[8] D. J. Bowersox, D. J. Class and M. B. Cooper. *Supply Chain Logistics Management*, Irwin/McGraw Hill, 2002.

[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*, pages 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*, pages 401–408, 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*, pages 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, March 2001. More detailed version: Technical Report CS-TR-3987, University of Maryland, February 1999, available at `http://www.cs.umd.edu/~dekhtyar/academ/publications/tp-final.ps`.

[14] D.J. Dewitt, N. Kabra, J. Luo, J. Patel and J. Yu. Client-Server Paradise. In *Proceedings 20th VLDB Conference*, Santiago, Chile, pages 558–569.

[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 of the 5th International Conference on Data Engineering (ICDE-89)*, pages 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, September 2001.

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

[21] S. Gadia, S. Nair, and Y. C. Poon. Incomplete information in relational temporal databases. In *Proceedings of the 18th International Conference on Very Large Databases (VLDB-92)*, pages 395–406, 1992.

[22] 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.

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

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

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

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

[27] M. Koubarakis. Complexity results for first order theories of temporal constraints. In *Proceeding of the 4th International Conference on Knowledge Representation and Reasoning (KR-94)*, pages 379–390, 1994.

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

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

[30] 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/`.

[31] 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.

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

[33] 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.

[34] 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.

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

[36] H.-J. Schek and P. Pistor. Data structures for an integrated data base management and information retrieval system. In *Proceedings of the 8th International Conference on Very Large Data Bases*, pages 197–207. Morgan Kaufmann, 1982.

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

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

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

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

[41] 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*, pages 201–209. ACM Press, 1994.

[42] 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 and Addison-Wesley, 1995.

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

[44] S. L. Vandenberg and D. J. DeWitt. Algebraic support for complex objects with arrays, identity, and inheritance. In *Proceedings SIGMOD-91*, pages 158–167. ACM Press, 1991.