Technical Report 29-01, December 2001.
Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", ITALY.
URL = http://www.dis.uniroma1.it/pub/AI/papers/tr-29-01.pdf

Temporal Probabilistic Object Bases

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

Abstract

There are numerous applications where we know that a certain event occurred during some time period, but we do not know exactly when that event occurred. Dyreson and Snodgrass have shown how this kind of temporal uncertainty can be handled in relational databases. In this paper, we propose two data models to handle temporal indeterminacy in object bases. The first model, which we call the *explicit* model, provides an extension of the relational algebra that explicitly considers all possibilities. This makes defining algebraic operations easy, but makes their implementation quite inefficient. The second model, which we call the *implicit* model, overcomes these deficiencies by proposing the intelligent use of constraints. This causes the model to be succinct. We also propose an *implicit* algebra on the implicit representation. We show that each implicit algebra operation precisely captures its explicit counterpart.

1 Introduction

There are numerous applications involving temporal indeterminacy. For instance, consider a commercial package delivery company (examples of companies in this broad class include UPS, Fedex, DHL, and many others). Such a company 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 statistical information about when the deliveries will arrive (an answer of the form "There is a 10 - 20% probability of the package being delivered between 9 am and 1pm, and a 80 - 90% probability of being delivered between 1pm and 5pm") is far more helpful to the company's decision making processes than the bland answer given today ("It will be delivered sometime today between 9 am and 5pm"). Temporal indeterminacy also arises in many other situations. Dyreson and Snodgrass [5] have identified numerous other applications where temporal indeterminacy is important. For example, radio-carbon dating efforts in archaeology are temporally indeterminate — a historical relic may be dated as "sometime between 500 and 400 BC." Likewise, time-series prediction programs are also uncertain about when certain events will occur. There are literally hundreds of stock market prediction programs containing models of when stocks are expected to reach certain prices. When the results of such programs are stored in databases and subjected to querying, the need to handle temporal indeterminacy is even more acute.

In this paper, we propose for the first time, a formal theoretical foundation for object bases containing temporal indeterminacy. As probabilities are the best known method for handling uncertain information, our model for indeterminacy (like that of Dyreson and Snodgrass [5]) is probabilistic. The organization and contributions of this paper are as follows.

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

†Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Via Salaria 113, 00198 Roma, Italy. E-mail: lukasiew@dis.uniroma1.it.

‡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.

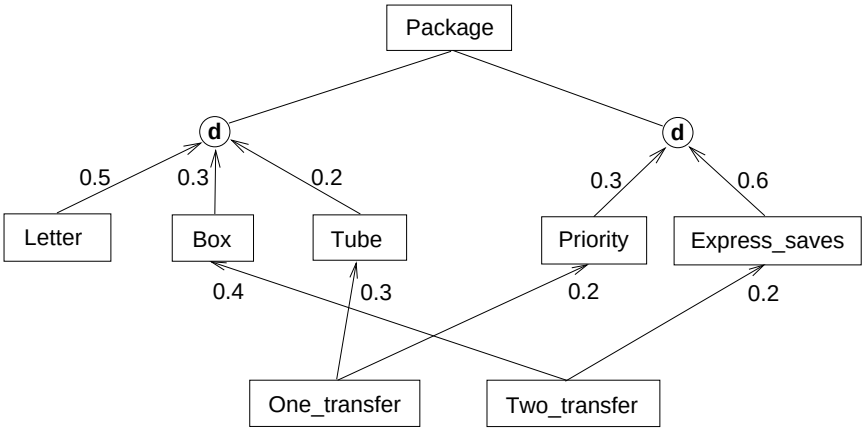

Figure 1: Package Example with probability assignment $\wp$

- In Section 2, we introduce some basic definitions in probability theory and temporal databases. An important definition introduced here is that of explicit values and implicit values — the latter are succinct representations of the former.
- In Section 3, we define the concept of a temporal probabilistic object base (TPOB for short). We define the important concept of an explicit TPOB-instance and an implicit TPOB-instance with the latter being a succinct representation of the former.
- In Section 4, we describe an *explicit TPOB-algebra* (*e-algebra* for short) that operates on explicit TPOB-instances. The advantage of the e-algebra is that using it, it is relatively intuitive to define the operations. However, it does not have an efficient implementation.
- In Section 5, we define an *implicit TPOB-algebra* (*i-algebra* for short) that operates on implicit TPOB-instances. We show that the i-algebra *correctly* implements the e-algebra (in other words, the answers produced by the i-algebra operations succinctly represent the answers produced by the corresponding e-algebra operations). As the i-algebra works on succinct representations, it has better computational properties.
- In Section 6, we compare our work with related work on temporal indeterminacy.
- Section 7 contains directions for future work and concludes the paper.

The key contributions of this paper are: (1) the definition of implicit TPOB-instances, (2) the definition of the implicit algebra operations, and (3) the results stating that these *implicit* algebra operations are correct implementations of the *explicit* algebra operations. The importance of (3) cannot be overemphasized — for example, a simple statement such as "Package P will be delivered sometime between 9 am and 5 pm today" expands out into 8 explicit statements if our chronon (i.e., smallest temporal granularity used) is hour, 480 explicit statements if our chronon is minute, and 28,800 explicit statements if our chronon is a second. Thus, a single statement in the implicit algebra can capture huge amounts of explicit data in a succinct fashion. The ability to correctly manipulate this implicit data is critical to the efficiency of TPOB systems.

Our work builds directly on top of pioneering work by Dyreson and Snodgrass [5]. Our work differs from theirs in several ways. First, it applies to object bases, while theirs applies to 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. 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, all our operations preserve coherence and consistency.

2 Basic Definitions

In this section, we recapitulate some basic definitions. In particular, we recall the notion of a calendar. We then define classical types and their values, and introduce probabilistic types and their explicit and implicit values. Finally, we describe the concept of a probabilistic strategy.

2.1 Calendars

We now recapitulate the concept of a calendar due to Kraus et al. [15]. A calendar consists of a linear temporal hierarchy of time units and a validity predicate specifying valid time points.

Definition 2.1 (time unit) A *time unit* consists of a *name* and a *time-value set.*

For example, the time unit named day has the time-value set $\{1, 2, \ldots, 31\}$.

Definition 2.2 (linear temporal hierarchy) A *linear temporal hierarchy* is a finite set of distinct time units with a linear order $\sqsupseteq$ among them.

For example, $day \sqsupseteq hour \sqsupseteq minute$ is a linear temporal hierarchy.

Definition 2.3 (time point) Let $H = T_1 \sqsupseteq \ldots \sqsupseteq T_n$ be a linear temporal hierarchy. A *time point* over H is a tuple $(t_1, \ldots, t_n)$, where each t_i is a time-value in the time-value set of T_i. We use $<_H$ to denote the usual lexicographic order on all time points over H, which is defined by $(t_1, \ldots, t_n) <_H (t'_1, \ldots, t'_n)$ iff some $i \in \{1, \ldots, n\}$ exists such that $t_j = t'_j$ for all $j \in \{1, \ldots, i-1\}$ and $t_i < t'_i$. We use $\leq_H$ to denote the reflexive closure of $<_H$.

For example, $(1997, 1, 31)$ is a time point over $H = year \sqsupseteq month \sqsupseteq day$.

Definition 2.4 (calendar) A *calendar* C consists of a linear temporal hierarchy H and a *validity predicate.* The validity predicate specifies a non-empty 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 assumed to be finite unless specified otherwise.

Intuitively, $(1997, 1, 31)$ and $(1997, 2, 31)$ are time points over $H = year \sqsupseteq month \sqsupseteq day$. The validity predicate may now characterize the former as valid and the latter as invalid. The reader interested in how to specify validity predicates may consult [15].

2.2 Types and Values

This section introduces types, and values associated with types. It is divided into three parts — classical types and values, probabilistic types, and explicit and implicit values of probabilistic types.

2.2.1 Classical Types and Values

Every classical type τ is associated with a *domain,* denoted $\mathrm{dom}(\tau)$, which specifies the set of values of τ. In this paper, we assume that $\mathcal{T} = \{\mathsf{integer}, \mathsf{string}, \mathsf{Boolean}, \mathsf{float}\}$ is the set of *atomic types* and that their domains are the usual domains. We also assume the existence of some arbitrary but fixed set $\mathcal{A}$ of attributes.

Definition 2.5 (temporal atomic type) Every calender τ is a *temporal atomic type* with the set of all valid time points as its associated domain dom(τ).

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

Definition 2.6 (classical type) We define *classical types* by induction as follows:

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

We may now define values of classical types.

Definition 2.7 (values of classical types) We define *values* of classical types inductively as follows:

- For all 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 $[A_1 \colon v_1, \ldots, A_k \colon v_k]$ is a value of the classical type $[A_1 \colon \tau_1, \ldots, A_k \colon \tau_k]$.

2.2.2 Probabilistic Types

A probabilistic type is either an atomic probabilistic type, or a complex probabilistic type constructed from classical types and atomic probabilistic types by using the tuple constructor.

Definition 2.8 (probabilistic type) We define *probabilistic types* by induction as follows:

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

Example 2.9 Consider an application maintaining information about how long it takes 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 atomic type string. In contrast, the attributes Delivery and STOPone may respectively be defined over the atomic probabilistic type $[\![\mathsf{time}]\!]$ and over the probabilistic tuple type $[\mathsf{City} \colon \mathsf{string}, \mathsf{Arrive} \colon \mathsf{string}, \mathsf{Shipment} \colon [\![\mathsf{time}]\!]]$, where time is a calendar.

Probabilistic types can have values that are represented either *explicitly* or *implicitly*. Each of these two options is now considered below.

2.2.3 Explicit Values of Probabilistic Types

We now introduce explicit values of probabilistic types.

Definition 2.10 (explicit values of probabilistic types) We define *explicit values* of probabilistic types by induction as follows:

- An explicit 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 $0 \leq l \leq u \leq 1$.
- An explicit 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 explicit values of $\tau_1, \ldots, \tau_k$.

Example 2.11 An example of an explicit value of the atomic probabilistic type $[\![\mathsf{time}]\!]$, where time is the calendar over the linear temporal hierarchy $hour \sqsupseteq minute$, is $\{((12, 30), [.4, .6]), ((12, 35), [.4, .6])\}$. An explicit value of the atomic probabilistic type $[\![\mathsf{string}]\!]$ is $\{(\mathsf{New\_York}, [.3, .4]), (\mathsf{Washington}, [.5, .6])\}$.

Let A be an attribute and $v = \{(v_1, [l_1, u_1]), \ldots, (v_n, [l_n, u_n])\}$ be an explicit value of an atomic probabilistic type. Then, "$A \colon v$" intuitively says that "The probability that A has the value v_i lies in the interval $[l_i, u_i]$". We assume that the events "$A \colon v_i$" are exhaustive and pairwise mutually exclusive, which implies the following notion of consistency for explicit values of probabilistic types.

Definition 2.12 (compactness and consistency) An explicit value $v = \{(v_1, [l_1, u_1]), \ldots, (v_n, [l_n, u_n])\}$ of an atomic probabilistic type is *compact* iff $v_1, \ldots, v_n$ are pairwise distinct. We say v is *consistent* iff v is compact and $\sum_{i=1}^{n} l_i \leq 1 \leq \sum_{i=1}^{n} u_i$. An explicit value v of a probabilistic type is *compact* (resp., *consistent*) iff all contained explicit values of atomic probabilistic types are compact (resp., consistent).

2.2.4 Implicit Values of Probabilistic Types

We now introduce implicit values of probabilistic types. These are implicit representations of explicit values — we generalize a constraint-based approach due to Dekhtyar et al. [2].

Temporal and Data Constraints. Using a temporal constraint, we can implicitly define a set of valid time points (namely the solutions of that constraint) w.r.t. a given calendar. In contrast, a data constraint specifies a set of data values from a totally ordered domain. We now define the syntax of temporal and data constraints.

Definition 2.13 (temporal constraint) Let τ be a calendar with linear temporal hierarchy $H = T_1 \sqsupseteq \cdots \sqsupseteq T_n$. An *atomic temporal constraint for* τ has one of the following forms:

- $(T_i\,\theta\,v_i)$ where θ belongs to $\{\leq, <, =, \neq, >, \geq\}$ and v_i is a time-value in the time-value set of time unit T_i. We call $(T_i\,\theta\,v_i)$ an *atomic time-value constraint.*
- $(t_1 \sim t_2)$ where $t_1, t_2 \in \mathrm{dom}(\tau)$ and $t_1 \leq_H t_2$. We call $(t_1 \sim t_2)$ an *atomic time-interval constraint.* We use (t_1) to abbreviate $(t_1 \sim t_1)$.

A *temporal constraint for* τ is a Boolean combination of atomic temporal constraints for τ (that is, constructed from atomic temporal constraints by using the Boolean operators $\neg$, $\wedge$, and $\vee$).

Definition 2.14 (data constraint) Let τ be a classical type with totally ordered domain $\mathrm{dom}(\tau)$. An *atomic data constraint for* τ is either of the form $(D\,\theta\,v)$, 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 (v_1) to abbreviate $(v_1 \sim v_1)$. A *data constraint for* τ is a Boolean combination of atomic data constraints for τ.

We now define the semantics of temporal and data constraints, that is, the set of time points and data values, respectively, that they specify.

Definition 2.15 (solution to a temporal constraint) Let τ be a calendar with linear temporal hierarchy $H = T_1 \sqsupseteq \cdots \sqsupseteq T_n$. A time point $s = (s_1, \ldots, s_n) \in \mathrm{dom}(\tau)$ is a *solution* to an atomic temporal constraint $(T_i\,\theta\,v_i)$ (resp., $(t_1 \sim t_2)$), denoted $s \models (T_i\,\theta\,v_i)$ (resp., $s \models (t_1 \sim t_2)$), iff $s_i\,\theta\,v_i$ (resp., $t_1 \leq_H s \leq_H t_2$). We inductively extend the notion of solutions to all temporal constraints by:

- $s \models \neg C_1$ iff it is not the case that $s \models C_1$,
- $s \models (C_1 \wedge C_2)$ iff $s \models C_1$ and $s \models C_2$,
- $s \models (C_1 \vee C_2)$ iff $s \models C_1$ or $s \models C_2$.

Definition 2.16 (solution to a data constraint) Let τ be any classical type with totally ordered domain $\mathrm{dom}(\tau)$. A value $s \in \mathrm{dom}(\tau)$ is a *solution* to $(D\,\theta\,v)$ (resp., $(v_1 \sim v_2)$), denoted $s \models (D\,\theta\,v)$ (resp., $s \models (v_1 \sim v_2)$), iff $s\,\theta\,v$ (resp., $v_1 \leq s \leq v_2$). The concept of solutions is extended in the usual way to all data constraints (see Definition 2.15).

Definition 2.17 (constraint and solution set) Let τ be a calendar (resp., classical type with totally ordered domain $\mathrm{dom}(\tau)$). A *constraint* C *for* τ is a temporal (resp., data) constraint for τ. We use $\mathrm{sol}(C)$ to denote the set of all solutions $s \in \mathrm{dom}(\tau)$ to C.

Probability Distribution Functions. We are now ready to recall the well-known concept of a probability distribution function (pdf for short). In the sequel, we assume that S is a set of $n \geq 1$ pairwise mutually exclusive events $s_1, \ldots, s_n$.

Definition 2.18 (probability distribution function) A *probability distribution function* (or simply *distribution function*) over S is a mapping $\rho\colon S \to [0,1]$ such that $\sum_{s \in S} \rho(s) \leq 1$.

The most widely used pdf is the uniform distribution: The *uniform distribution* over S, denoted U_S, 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 probability distribution functions. Here, we additionally assume that S is totally ordered by $s_i < s_j$ iff $i < j$ for all $i, j \in \{1, \ldots, n\}$:

- 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 $s_i \in S$.

Implicit Values of Probabilistic Types. In order to define implicit values of probabilistic types, we start by defining implicit tuples — a concept borrowed from [2].

Definition 2.19 (implicit tuple) Let τ be either a calendar or a classical type with a totally ordered domain. An *implicit tuple* (or *i-tuple*) for τ is a 5-tuple (C, D, l, u, δ), where C, D are constraints for τ with $\emptyset \subset \text{sol}(C) \subseteq \text{sol}(D)$, l, u are reals with $0 \leq l \leq u \leq 1$, and δ is a distribution function over $\text{sol}(D)$. If $\text{sol}(C) = \text{sol}(D)$, we use $(\#)$ to abbreviate C.

We next give a formal definition of implicit values of probabilistic types.

Definition 2.20 (implicit values of probabilistic types) We 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 $\tau_1, \ldots, \tau_k$.

Example 2.21 An implicit value of the atomic probabilistic type $[\![\mathsf{time}]\!]$, where time is the calendar over the linear temporal hierarchy $hour \sqsupseteq minute$, is $v = ((\#), ((12, 30) \sim (12, 34)), 0.5, 1, U)$.

Every implicit value v of an atomic probabilistic type $[\![\tau]\!]$ has an equivalent explicit value $\varepsilon(v)$, which is defined by $\varepsilon(v) = \{(v', [l \cdot \delta(v'), u \cdot \delta(v')]) \mid \exists (C, D, l, u, \delta) \in v \colon v' \in \text{sol}(C)\}$.

Example 2.22 Let us reconsider the implicit value v of Example 2.21. Its explicit value $\varepsilon(v)$ is given by $\{((12, 30), [.1, .2]), ((12, 31), [.1, .2]), ((12, 32), [.1, .2]), ((12, 33), [.1, .2]), ((12, 34), [.1, .2])\}$.

It is now easy to extend the notions of compactness and consistency to implicit values.

Definition 2.23 (compactness and consistency) 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 *compact* iff $\text{sol}(C_i \wedge C_j) \neq \emptyset$ for all $i, j \in \{1, \ldots, n\}$ with $i \neq j$. We say v is *consistent* iff v is compact and

$$\textstyle\sum_{i=1}^{n} \sum_{v_i \in \text{sol}(C_i)} l_i \cdot \delta_i(v_i) \quad \leq \quad 1 \quad \leq \quad \sum_{i=1}^{n} \sum_{v_i \in \text{sol}(C_i)} u_i \cdot \delta_i(v_i)\,.$$

An implicit value v of a probabilistic type is *compact* (resp., *consistent*) iff all contained implicit values of atomic probabilistic types are compact (resp., consistent).

2.3 Probabilistic Strategies

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 $e_1 \vee e_2$, $e_1 \wedge e_2$, 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 $e_1 \vee e_2$, $e_1 \wedge e_2$, and $e_1 \wedge \neg e_2$.

Definition 2.24 (conjunction, disjunction, and difference strategy) Let $\mathcal{U}$ denote 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., $e_1 \vee e_2$, $e_1 \wedge \neg e_2$).

Lakshmanan et al. [16] 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 1–3 show some examples of conjunction, disjunction, and difference strategies ([7] has 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., $[0, 0]$).

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	$([l_1, u_1] \ominus_{ig} [l_2, u_2]) = [\max(0, l_1 - u_2), \min(u_1, 1 - l_2)]$

3 Temporal Probabilistic Object Bases

In this section, we first introduce the concept of a schema for temporal probabilistic object bases. We then define the inheritance completion of a schema. Finally, we introduce temporal probabilistic object base instances with respect to this inheritance completion.

3.1 Temporal Probabilistic Object Base Schema

A temporal probabilistic object base schema consists of a hierarchy of classes with associated types. Membership of an object in an immediate subclass of any class is expressed by a conditional probability value.

Definition 3.1 (temporal probabilistic object base schema) A *temporal probabilistic object base schema (TPOB-schema)* is a 5-tuple $(\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$, where:

- $\mathcal{C}$ is a finite set of *classes.*
- σ maps each class $c \in \mathcal{C}$ to a probabilistic tuple type. Intuitively, σ specifies the data type of each class.
- $\Rightarrow$ is a binary relation on $\mathcal{C}$ such that $(\mathcal{C}, \Rightarrow)$ is a directed acyclic graph (dag). Intuitively, each node of the directed acyclic graph $(\mathcal{C}, \Rightarrow)$ is a class from $\mathcal{C}$, and each edge $c_1 \Rightarrow c_2$ says that the class c_1 is an *immediate subclass* of c_2.
- me maps each class c to a partition of the set of all immediate subclasses of c. Intuitively, suppose that the class c has the five subclasses $c_1, \ldots, c_5$ and suppose that $\mathrm{me}(c)$ is the partition $\{\{c_1, c_2\}, \{c_3, c_4, c_5\}\}$. Here, $\mathrm{me}(c)$ produces two clusters. An object o belonging to class c can belong to no, or either, or both clusters of c. However, the classes within a cluster are mutually exclusive, that is, o cannot belong to both c_1 and c_2 in the same time.
- $\wp$ maps each edge in $(\mathcal{C}, \Rightarrow)$ to a positive rational number in the unit interval $[0, 1]$ such that for all classes c and all clusters $\mathcal{P} \in \mathrm{me}(c)$, it holds that $\Sigma_{d \in \mathcal{P}} \wp(d, c) \leq 1$. Intuitively, if $c_1 \Rightarrow c_2$, then $\wp(c_1, c_2)$ specifies the conditional probability that an arbitrary object belongs to the subclass c_1 given that it belongs to the superclass c_2.

Example 3.2 The TPOB-schema for the Package Example contains the following components:

- $\mathcal{C} = \{\mathsf{Package}, \mathsf{Letter}, \mathsf{Box}, \mathsf{Tube}, \mathsf{Priority}, \mathsf{Express\_saves}, \mathsf{One\text{-}transfer}, \mathsf{Two\text{-}transfer}\}$.

- σ is given by Table 4 below.
- $(\mathcal{C}, \Rightarrow)$ is the graph resulting from Figure 1 when the d-nodes are contracted to Package and the probability labels are removed.
- $\mathrm{me}(\mathsf{Package}) = \{\{\mathsf{Letter}, \mathsf{Box}, \mathsf{Tube}\}, \{\mathsf{Priority}, \mathsf{Express\_saves}\}\}$,
  $\mathrm{me}(\mathsf{Box}) = \mathrm{me}(\mathsf{Express\_saves}) = \{\{\mathsf{Two\text{-}transfer}\}\}$,
  $\mathrm{me}(\mathsf{Tube}) = \mathrm{me}(\mathsf{Priority}) = \{\{\mathsf{One\text{-}transfer}\}\}$,
  $\mathrm{me}(\mathsf{Letter}) = \mathrm{me}(\mathsf{One\text{-}transfer}) = \mathrm{me}(\mathsf{Two\text{-}transfer}) = \emptyset$.
- $\wp$ is the probability assignment in Figure 1.

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]
Tube	[Height: float, Width: float, Depth: float]
Priority	[Time: [[time]]]
Express_saves	[Time: [[time]]]
One-transfer	[City: string, Arrive: [[time]], Shipment: [[time]]]
Two-transfer	[STOPone: [City: string, Arrive: [[time]], Shipment: [[time]]], STOPtwo: [City: string, Arrive: [[time]], Shipment: [[time]]]]

A *directed path* in the dag $(\mathcal{C}, \Rightarrow)$ is a sequence of classes $c_1, c_2, \ldots, c_k$ such that $c_1 \Rightarrow c_2 \Rightarrow \cdots \Rightarrow c_k$ and $k \geq 1$. We use $\Rightarrow^\star$ to denote the reflexive and transitive closure of $\Rightarrow$. We say that c_1 is a subclass of c_2 iff $c_1 \Rightarrow^\star c_2$. We say c_1 is a strict subclass of c_2, iff c_1 is a subclass of c_2 and $c_1 \neq c_2$. A class d is *minimal* under $\Rightarrow^\star$ in a set of classes $\mathcal{D}$ iff $d \in \mathcal{D}$ and no class in $\mathcal{D}$ is a strict subclass of d. A class d is a *subclass* of a partition cluster $\mathcal{P}$ iff d is a subclass of some $c \in \mathcal{P}$.

We are now ready to define a notion of consistency for TPOB-schemas.

Definition 3.3 (consistent TPOB-schema) Let $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ be a TPOB-schema. An *interpretation* of $\mathbf{S}$ is any mapping ε from $\mathcal{C}$ to the set of all finite subsets of a set $\mathcal{O}$. An interpretation ε of $\mathbf{S}$ is called a *taxonomic model* of $\mathbf{S}$ iff it satisfies the following conditions:

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

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

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

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

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

We say $\mathbf{S}$ is *consistent* iff $\mathbf{S}$ has a model.

The work in this section builds upon definitions in [6]. 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 [6] for detailed algorithms).

3.2 Inheritance Completion

We now define the inheritance completion of a TPOB-schema. Intuitively, a class in a TPOB-schema should inherit attributes from its ancestors. An inheritance strategy, defined below, explains how to resolve conflicts that arise due to multiple inheritance.

Definition 3.4 (inheritance strategy) Let $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ be a TPOB-schema. Let $\boldsymbol{A}$ denote the set of all top-level attributes of $\mathbf{S}$. Let $\mathsf{min}_{\mathbf{S}} : \mathcal{C} \times \boldsymbol{A} \rightarrow \mathcal{C}$ be the mapping that assigns to each pair $(c, A) \in \mathcal{C} \times \boldsymbol{A}$ the set of all classes $d \in \mathcal{C}$ such that (i) $c \Rightarrow^\star d$, (ii) A is a top-level attribute of d, and (iii) no other d' such that $c \Rightarrow^\star d' \Rightarrow^\star d$ satisfies conditions (i) and (ii). An *inheritance strategy* for $\mathbf{S}$ is a partial mapping $\mathsf{inh}_{\mathbf{S}} : \mathcal{C} \times \boldsymbol{A} \rightarrow \mathcal{C}$ that assigns a class $d \in \mathsf{min}_{\mathbf{S}}(c, A)$ to each $(c, A) \in \mathcal{C} \times \boldsymbol{A}$ 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 superclasses.

Definition 3.5 (inheritance completion TPOB-schema) The *inheritance completion TPOB-schema* of a TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ is the TPOB-schema $\mathbf{S}^\star = (\mathcal{C}, \sigma^\star, \Rightarrow, \mathrm{me}, \wp)$, where $\sigma^\star(c) = [A_1 : \tau_1, \ldots, A_k : \tau_k]$ such that some $d \in \mathcal{C}$ exists with (i) $\mathsf{inh}_{\mathbf{S}}(c, A_i) = d$ and (ii) A_i is a top-level attribute of d with type τ_i. If $\mathbf{S} = \mathbf{S}^\star$, then $\mathbf{S}$ is said to be *fully inherited.*

Example 3.6 Consider the TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ given in Example 3.2. Its inheritance completion TPOB-schema is given by $\mathbf{S}^\star = (\mathcal{C}, \sigma^\star, \Rightarrow, \mathrm{me}, \wp)$, where $\sigma^\star$ is given in Table 5.

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]
Tube	[Origin: string, Destination: string, Delivery: [[time]], Height: float, Width: float, Depth: float]
Priority	[Origin: string, Destination: string, Delivery: [[time]], Time: [[time]]]
Express_saves	[Origin: string, Destination: string, Delivery: [[time]], Time: [[time]]]
One-transfer	[Origin: string, Destination: string, Delivery: [[time]], Height: float, Width: float, Depth: float, Time: [[time]], City: string, Arrive: [[time]], Shipment: [[time]]]
Two-transfer	[Origin: string, Destination: string, Delivery: [[time]], Height: float, Width: float, Depth: float, Time: [[time]], STOPone: [City: string, Arrive: [[time]], Shipment: [[time]]], STOPtwo: [City: string, Arrive: [[time]], Shipment: [[time]]]]

3.3 Temporal Probabilistic Object Base Instance

In this section, we introduce the notion of a temporal probabilistic object base instance, which is defined with respect to the inheritance completion of a TPOB-schema.

Assumption. In the rest of this paper, we assume that there is a (countably) infinite set $\mathcal{O}$ of *object identifiers* (*oids*). For the algebraic operations of natural join, Cartesian product, and conditional join (see Section 4), we assume that $\mathcal{O} \times \mathcal{O} \subseteq \mathcal{O}$, that is, $\mathcal{O}$ is closed under Cartesian product.

A temporal probabilistic object base instance associates with each class c a set of oids $\pi(c)$, and with each oid $o \in \pi(c)$ a value of appropriate type $\sigma^\star(c)$.

Definition 3.7 (temporal probabilistic object base instance) A *temporal probabilistic object base instance* (*TPOB-instance*) $\mathbf{I}$ over a consistent TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ is a pair (π, ν), where (i) $\pi : \mathcal{C} \rightarrow 2^{\mathcal{O}}$ maps each class c to a finite subset of $\mathcal{O}$ such that $\pi(c_1) \cap \pi(c_2) = \emptyset$ for any distinct $c_1, c_2 \in \mathcal{C}$, and (ii) ν maps each oid $o \in \pi(c)$, $c \in \mathcal{C}$, to a value of type $\sigma^\star(c)$. We say $\mathbf{I}$ is *explicit* (resp., *implicit*) iff all $\nu(o)$ with $o \in \pi(c)$ for some $c \in \mathcal{C}$ are explicit (resp., implicit) values. Every implicit instance $\mathbf{I}$ is

associated with the explicit instance $\varepsilon(\mathbf{I})$, which is obtained from $\mathbf{I}$ by replacing every contained implicit value v of an atomic probabilistic type by its corresponding explicit value $\varepsilon(v)$.

We define $\pi(\mathcal{C}) = \bigcup\{\pi(c) \mid c \in \mathcal{C}\}$ and $\pi^\star(c) = \bigcup\{\pi(c') \mid c' \in \mathcal{C}, c' \Rightarrow^\star c\}$. Informally, $\pi(c)$ is the set of all objects that are *created* in c, while $\pi^\star(c)$ is the set of all objects that *belong* to c.

Example 3.8 Tables 6 and 7 show a sample TPOB-instance over the TPOB-schema of Example 3.2.

Table 6: The mappings π and π^*

c	$\pi(c)$	$\pi^*(c)$
Package	$\{\}$	$\{o_3, o_5\}$
Letter	$\{\}$	$\{\}$
Box	$\{\}$	$\{o_5\}$
Tube	$\{\}$	$\{\}$
Priority	$\{o_3\}$	$\{o_3\}$
Express_saves	$\{\}$	$\{o_5\}$
One-transfer	$\{\}$	$\{\}$
Two-transfer	$\{o_5\}$	$\{o_5\}$

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

We now define the concept of a probabilistic extent. Informally, the probabilistic extent of a class $c \in \mathcal{C}$ specifies the probability that an oid $o \in \pi(\mathcal{C})$ belongs to c.

Definition 3.9 (probabilistic extent) Let $\mathbf{I} = (\pi, \nu)$ be a TPOB-instance over a consistent TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$, 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 (that is, for all models ε of $\mathbf{S}$, the sets $\varepsilon(c')$ and $\varepsilon(c)$ are disjoint), then $\mathrm{ext(c)}(o) = \{0\}$.

(3) Otherwise, $\mathrm{ext(c)}(o) = \{p \mid p$ is the product of the edge probabilities on a path from c up to a class d such that (i) $o \in \pi^\star(d)$ and $c \Rightarrow^\star d$, and (ii) d is minimal under $\Rightarrow^\star$ with (i)$\}$.

We call the class c in (1), the class c' in (2), and every class d that satisfies (i) and (ii) in (3) a *characteristic class for* $\mathrm{ext(c)}(o)$.

Example 3.10 The probabilistic extent of One-transfer in our Package Example is as follows: $\mathrm{ext(One\text{-}transfer)}(o_3) = \{.2\}$, $\mathrm{ext(One\text{-}transfer)}(o_5) = \{0\}$.

The notion of coherence of TPOB-instances given below requires that the probabilistic extent of every class assigns a unique probability to every oid.

Definition 3.11 (coherent TPOB-instance) Let $\mathbf{I} = (\pi, \nu)$ be a TPOB-instance over a consistent TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$. The TPOB-instance $\mathbf{I}$ is *coherent* iff for all classes $c \in \mathcal{C}$ and all objects $o \in \pi(\mathcal{C})$, the probabilistic extent $\mathrm{ext(c)}(o)$ contains *at most* one element.

We now come to the important notion of consistency for TPOB-instances.

Definition 3.12 (consistent TPOB-instance) A TPOB-instance $\mathbf{I} = (\pi, \nu)$ over a TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ is *consistent* if $\nu(o)$ is consistent for all $o \in \pi(\mathcal{C})$.

4 Explicit Temporal Probabilistic Object Algebra

In this section, we introduce the explicit TPOB-algebra (e-algebra). The e-algebra operators take explicit TPOB-instances over TPOB-schemas as input and produce an explicit TPOB-instance over a TPOB-schema as output. *Unless specified otherwise, we assume that all input TPOB-schemas are fully inherited and that all input TPOB-instances are explicit.*

4.1 Selection

We first define the selection operation with respect to probabilistic selection conditions. We start by defining path expressions and atomic selection conditions, which are then used to construct selection conditions and probabilistic selection conditions.

Definition 4.1 (path expression) We define *path expressions* inductively as follows:

- A path expression for the classical or probabilistic tuple type $[A_1 : \tau_1, \ldots, A_k : \tau_k]$ is either of the form A_i or of 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.

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

Definition 4.2 (atomic selection condition) Let $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ be a TPOB-schema. 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 $\theta \in \{\leq, <, =, \neq, >, \geq\}$.

For example, $in(\mathsf{Letter})$ and $\mathsf{STOPone.City} = \mathsf{New\_York}$ are two atomic selection conditions. They say "find all objects that are letters" and "find all objects that have New York as the first stop", respectively.

Definition 4.3 (selection condition) We define *selection conditions* by induction as follows.

- Every atomic selection condition is a selection condition.
- 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.

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

Definition 4.4 (probabilistic selection condition) 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. If ϕ and ψ are probabilistic selection conditions, then so are $\neg\phi$, $(\phi \wedge \psi)$, and $(\phi \vee \psi)$.

For instance, $(\mathsf{Time} < (14, 00))[.5, 1]$ is a probabilistic selection condition, which says "find all objects whose values in the attribute Time are smaller than 14:00 with associated probability in $[.5, 1]$".

In order to define the semantics of probabilistic selection conditions, we first define the semantics of path expressions.

Definition 4.5 (valuation of path expressions) Let P be a path expression for the probabilistic type τ. The *valuation* of P under an explicit value v of τ, denoted $v.P$, is defined as follows:

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

We may now define the semantics of atomic selection conditions and selection conditions.

Definition 4.6 (valuation of atomic selection conditions) Let $\mathbf{I}{=}(\pi, \nu)$ be a TPOB-instance over a 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. The *probabilistic valuation* with respect to $\mathbf{I}$ and o, denoted $\mathrm{prob}_{\mathbf{I},o}$, is the following partial mapping from the set of all atomic selection conditions to the set of all closed subintervals of $[0, 1]$:

- $\mathrm{prob}_{\mathbf{I},o}(in(c)) = [\min(\mathrm{ext(c)}(o)), \max(\mathrm{ext(c)}(o))]$.
- 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 explicit 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 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 explicit 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.

In the Package Example, the atomic selection condition $in(\mathsf{One\text{-}transfer})$ is assigned the intervals $[.2, .2]$ and $[0, 0]$ under $\mathrm{prob}_{\mathbf{I},o_3}$ and $\mathrm{prob}_{\mathbf{I},o_5}$, respectively. The atomic selection condition $\mathsf{STOPone.City} = \mathsf{New\_York}$ is undefined under $\mathrm{prob}_{\mathbf{I},o_3}$, and it is assigned $[1, 1]$ under $\mathrm{prob}_{\mathbf{I},o_5}$.

Definition 4.7 (valuation of selection conditions) Let $\mathbf{I}{=}(\pi, \nu)$ be a TPOB-instance over a TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$, and let $o \in \pi(\mathcal{C})$. We extend $\mathrm{prob}_{\mathbf{I},o}$ to a partial mapping from the set of all selection conditions to the set of all closed subintervals of $[0, 1]$ as follows:

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

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

We are now ready to define what it means for an object o in a TPOB-instance $\mathbf{I}$ over some TPOB-schema $\mathbf{S}$ to satisfy a probabilistic selection condition ϕ.

Definition 4.8 (satisfaction of probabilistic selection conditions) Let $\mathbf{I} = (\pi, \nu)$ be a TPOB-instance over a TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$, and let $o \in \pi(\mathcal{C})$. Let ϕ be a probabilistic selection condition such that $\mathrm{prob}_{\mathbf{I},o}(\phi_s)$ is defined for all selection conditions ϕ_s that occur in ϕ. The *satisfaction* of ϕ under I and o, denoted $\mathrm{prob}_{\mathbf{I},o} \models \phi$, is inductively defined as follows:

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

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

We illustrate the notion of satisfaction via the following example.

Example 4.9 The following satisfaction (and non-satisfaction) relations hold in our Package Example.

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

We are now ready to define the selection operation on TPOB-instances.

Definition 4.10 (selection on TPOB-instances) Let $\mathbf{I} = (\pi, \nu)$ be a TPOB-instance over a TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \text{me}, \wp)$, and let ϕ be a probabilistic selection condition. The *selection on* $\mathbf{I}$ *with respect to* ϕ, denoted $\sigma_\phi(\mathbf{I})$, is the TPOB-instance (π', ν') over $\mathbf{S}$, where:

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

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

Example 4.11 Let $\mathbf{I} = (\pi, \nu)$ be the TPOB-instance of Example 3.8, and let the probabilistic selection condition ϕ be given by $\neg(in(\mathsf{Priority}) \wedge_{\otimes_{in}} \mathsf{Time} < (14,00))[0,0]$. Then, $\sigma_\phi(\mathbf{I}) = (\pi', \nu')$, where π' is given by $\pi'(\mathsf{Priority}) = \{o_3\}$ and $\pi'(c) = \emptyset$ 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: $\{((8,00),[.45,.5]),((8,10),[.4,.5])\}$]

4.2 Restricted Selection

We now introduce the operation of restricted selection. Informally, this operation is a selection on the explicit values of an atomic probabilistic type of an object. Before we can define restricted selection on TPOB-instances, an intermediate definition is needed.

Definition 4.12 (restricted selection on explicit values) Let τ be a probabilistic tuple type. Let ϕ be of the form $P.C$, where P is a path expression for τ, and C is a constraint. Let v be an explicit value of τ. The *restricted selection on* v *with respect to* ϕ, denoted $\sigma^r_\phi(v)$, is defined by:

- If $v = [A_1 \colon v_1, \ldots, A_i \colon v_i, \ldots, A_k \colon v_k]$, v_i is a value of a classical type, $P = A_i$, and $v_i \in \text{sol}(C)$, then $\sigma^r_\phi(v) = v$.
- If $v = [A_1 \colon v_1, \ldots, A_i \colon v_i, \ldots, A_k \colon v_k]$, v_i is an explicit value of an atomic probabilistic type, and $P = A_i$, then $\sigma^r_\phi(v) = [A_1 \colon v_1, \ldots, A_i \colon \{(v_i', I) \in v_i \mid v_i' \in \text{sol}(C)\}, \ldots, A_k \colon v_k]$.
- If $v = [A_1 \colon v_1, \ldots, A_i \colon v_i, \ldots, A_k \colon v_k]$, v_i is an explicit value of a probabilistic tuple type, and $P = A_i.R$, then $\sigma^r_\phi(v) = [A_1 \colon v_1, \ldots, A_i \colon \sigma^r_{R.C}(v_i), \ldots, A_k \colon v_k]$.
- Otherwise, $\sigma^r_\phi(v)$ is undefined.

We are now ready to define the restricted selection operator on TPOB-instances.

Definition 4.13 (restricted selection on TPOB-instances) Let $\mathbf{I}=(\pi, \nu)$ be a TPOB-instance over a TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \text{me}, \wp)$. Let ϕ be an expression of the form $P.C$, where P is a path expression for all $\sigma(c)$ with $c \in \mathcal{C}$, and C is a constraint. 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 4.14 Let $\mathbf{I} = (\pi, \nu)$ be the TPOB-instance given in Example 4.11, and let $\phi = (\mathsf{Time} < (8, 05))$. Then, $\sigma^r_\phi(\mathbf{I}) = (\pi', \nu')$, 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: $\{((8,00),[.45,.5])\}$]

4.3 Renaming

We now define the renaming operation. Informally, this operation renames some attributes in types of TPOB-schemas and 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.

Definition 4.15 (renaming condition) Let τ be a probabilistic tuple type. A *renaming condition* for 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 4.16 Let τ be the probabilistic tuple type [STOPone: [City: string, Arrive: $[\![$time$]\!]$, Shipment: $[\![$time$]\!]$]]. A renaming condition for τ is given by STOPone.City, STOPone$\leftarrow$STOPone.City1, STOPone1.

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.

Definition 4.17 (renaming on probabilistic tuple types) Let N be a renaming condition of the form $P \leftarrow P'$ for the probabilistic tuple type $\tau = [A_1 \colon \tau_1, \ldots, A_n \colon \tau_n]$. 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.

Definition 4.18 (renaming of TPOB-schemas) 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 $c \in \mathcal{C}$.

Before defining the renaming on TPOB-instances, we need to define it on explicit values.

Definition 4.19 (renaming of explicit values) 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 explicit value of τ. The *renaming* of 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 $(v_i', I_i) \in v_i$ 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 finally ready to define the renaming of TPOB-instances.

Definition 4.20 (renaming of TPOB-instances) 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 $o \in \pi(\mathcal{C})$.

4.4 Projection

The projection operation removes some attributes with their associated types from TPOB-schemas, and the same attributes with their values from TPOB-instances. We first define the projection of TPOB-schemas on a set of attributes.

Definition 4.21 (projection of TPOB-schemas) Let $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ be a TPOB-schema, and let $\mathbf{A}$ be a set of attributes. The *projection of* $\mathbf{S}$ *on* $\mathbf{A}$, denoted $\Pi_{\mathbf{A}}(\mathbf{S})$, is defined as the TPOB-schema $(\mathcal{C}, \sigma', \Rightarrow, \mathrm{me}, \wp)$, where the new type $\sigma'(c)$ of each class $c \in \mathcal{C}$ is obtained from the old type $\sigma(c) = [B_1 : \tau_1, \ldots, B_k : \tau_k]$ by removing all $B_i : \tau_i$'s with $B_i \notin \mathbf{A}$.

We may now define the projection of a TPOB-instance on a set of attributes.

Definition 4.22 (projection of TPOB-instances) Let $\mathbf{I} = (\pi, \nu)$ be a TPOB-instance over a TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$, and let $\mathbf{A}$ be a set of attributes. The *projection of* $\mathbf{I}$ *on* $\mathbf{A}$, denoted $\Pi_{\mathbf{A}}(\mathbf{I})$, is defined as the 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 $B_i \notin \mathbf{A}$.

Example 4.23 Consider the fully inherited TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ of Example 3.6 and the TPOB-instance $\mathbf{I} = (\pi, \nu)$ over $\mathbf{S}$ of Example 3.8. The assignments σ' and ν' obtained by projecting $\mathbf{S}$ and $\mathbf{I}$ on $\mathbf{A} = \{\mathsf{Origin}, \mathsf{Time}\}$ are shown in Tables 10 and 11, respectively.

Table 10: σ' resulting from projection

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

4.5 Extraction

To our knowledge, the extraction operation is one that has never been defined in probabilistic databases. This operation is unique to object bases because it allows for classes to be selected (and hence for other classes to be dropped) from the class hierarchy of a TPOB-schema.

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, Time: $\{((12,00),[.3,.4]),((12,05),[.4,.7])\}$]

That is, the extraction operation removes classes from the class hierarchy and all objects in the dropped classes. We first define the extraction operation on TPOB-schemas.

Definition 4.24 (extraction on TPOB-schemas) Let $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ be a TPOB-schema, and let $\mathbf{C}$ be a subset of $\mathcal{C}$. The *extraction on* $\mathbf{S}$ *with respect to* $\mathbf{C}$, denoted $\Xi_{\mathbf{C}}(\mathbf{S})$, is defined as the TPOB-schema $\mathbf{S}' = (\mathcal{C}', \sigma', \Rightarrow', \mathrm{me}', \wp')$, where

- $\mathcal{C}' = \mathbf{C}$.
- σ' is the restriction of σ to $\mathcal{C}'$.
- $\Rightarrow'$ is a binary relation on $\mathcal{C}'$ such that for each $c_1, c_2 \in \mathcal{C}'$, it holds $c_1 \Rightarrow' c_2$ iff there exists some path $d_1 \Rightarrow d_2 \Rightarrow \cdots \Rightarrow d_k$ such that $d_1 = c_1$, $d_k = c_2$, and $d_2, \ldots, d_{k-1} \in \mathcal{C} - \mathcal{C}'$.
- $\mathrm{me}'(c) = \{((X \cap \mathcal{C}') \cup X_{\mathcal{C}'}) \neq \emptyset \mid X \in \mathrm{me}(c)\}$, where $X_{\mathcal{C}'} = \{d_1 \in \mathcal{C}' \mid d_1 \Rightarrow d_2 \Rightarrow \cdots \Rightarrow d_k$ for some $d_2, \ldots, d_{k-1} \in \mathcal{C} - \mathcal{C}',\ d_k \in X - \mathcal{C}'\}$, for all $c \in \mathcal{C}'$.
- 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 4.25 Consider the fully inherited TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ of Examples 3.6. The extraction on $\mathbf{S}$ with respect to the set of classes $\mathbf{C} = \{$Package, Letter, Priority, One-transfer$\}$ is given by the TPOB-schema $\Xi_{\mathbf{C}}(\mathbf{S}) = (\mathcal{C}', \sigma', \Rightarrow', \mathrm{me}', \wp')$, where:

- $\mathcal{C}' = \{$Package, Letter, Priority, One-transfer$\}$.
- σ' is given in Table 12.
- $(\mathcal{C}', \Rightarrow')$ and $\wp'$ are given in Figure 2.
- $\mathrm{me}'($Package$) = \{\{$Letter, One-transfer$\}, \{$Priority$\}\}$, $\mathrm{me}'($Priority$) = \{\{$One-transfer$\}\}$, and $\mathrm{me}'($Letter$) = \mathrm{me}'($One-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, Time: [[time]], City: string, Arrive: [[time]], Shipment: [[time]]]

Our next step is to define the extraction operation on TPOB-instances.

Definition 4.26 (extraction on TPOB-instances) Let $\mathbf{I}=(\pi, \nu)$ be a TPOB-instance over a TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$, and let $\mathbf{C}$ be a set of classes from $\mathcal{C}$. The *extraction on* $\mathbf{I}$ *with respect to* $\mathbf{C}$, denoted $\Xi_{\mathbf{C}}(\mathbf{I})$, is the TPOB-instance (π', ν') over the TPOB-schema $\Xi_{\mathbf{C}}(\mathbf{S}) = (\mathcal{C}', \sigma', \Rightarrow', \mathrm{me}', \wp')$, 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 application of the extraction operator on TPOB-instances.

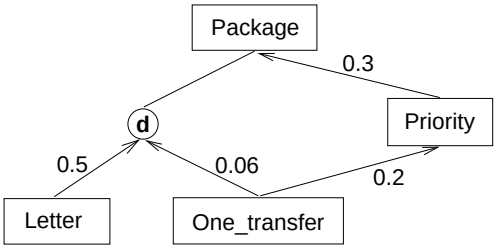

Figure 2: Class hierarchy and probability assignment resulting from extraction

Example 4.27 Consider the fully inherited TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ of Example 3.6 and the TPOB-instance $\mathbf{I} = (\pi, \nu)$ over $\mathbf{S}$ of Example 3.8. The extraction on $\mathbf{I}$ with respect to the set of classes $\mathbf{C} = \{\mathsf{Package}, \mathsf{Letter}, \mathsf{Priority}, \mathsf{One\text{-}transfer}\}$ results in a TPOB-instance (π', ν') over the TPOB-schema $\Xi_{\mathbf{C}}(\mathbf{S})$ in Example 4.25, 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	$\{\}$	$\{\}$

4.6 Natural Join

The operations we have presented thus far access only one TPOB-instance at a time. We now define the important concept of a natural join. Recall that for each class $c \in \mathcal{C}$ of a TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$, the type $\sigma(c)$ is a probabilistic tuple type over $\mathbf{S}$. Moreover, each oid $o \in \pi(c)$ that occurs in a TPOB-instance $\mathbf{I} = (\pi, \nu)$ over $\mathbf{S}$ is associated with a value of the probabilistic tuple type $\sigma(c)$. Each such o may be written as the list of the values (possibly complex values) for the top-level attributes $A_1, \ldots, A_m$ of $\sigma(c)$. We first describe when two TPOB-schemas $\mathbf{S}_1$ and $\mathbf{S}_2$ can be combined using natural join.

Definition 4.28 (natural-join-compatible TPOB-schemas) The TPOB-schemas $\mathbf{S}_1 = (\mathcal{C}_1, \sigma_1, \Rightarrow_1, \mathrm{me}_1, \wp_1)$ and $\mathbf{S}_2 = (\mathcal{C}_2, \sigma_2, \Rightarrow_2, \mathrm{me}_2, \wp_2)$ are *natural-join-compatible* iff for all classes $c_1 \in \mathcal{C}_1$ and $c_2 \in \mathcal{C}_2$, the common top-level attributes of $\sigma_1(c_1)$ and $\sigma_2(c_2)$ are associated with the same types in $\sigma_1(c_1)$ and $\sigma_2(c_2)$.

We may now define the natural join of two natural-join-compatible TPOB-schemas.

Definition 4.29 (natural join of TPOB-schemas) Let $\mathbf{S}_1 = (\mathcal{C}_1, \sigma_1, \Rightarrow_1, \mathrm{me}_1, \wp_1)$ and $\mathbf{S}_2 = (\mathcal{C}_2, \sigma_2, \Rightarrow_2, \mathrm{me}_2, \wp_2)$ be two natural-join-compatible TPOB-schemas. The *natural join* of $\mathbf{S}_1$ and $\mathbf{S}_2$, denoted $\mathbf{S}_1 \bowtie \mathbf{S}_2$, is the TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$, where

- $\mathcal{C} = \mathcal{C}_1 \times \mathcal{C}_2$.
- For all $c = (c_1, c_2) \in \mathcal{C}$, the probabilistic tuple type $\sigma(c) = [A_1 : \tau_1, \ldots, A_l : \tau_l]$ contains exactly all $A_i : \tau_i$ that belong to either the type $\sigma_1(c_1)$ or the type $\sigma_2(c_2)$.
- The directed acyclic graph $(\mathcal{C}, \Rightarrow)$ is defined as follows. For all $c = (c_1, c_2), d = (d_1, d_2) \in \mathcal{C}$: $c \Rightarrow d$ iff $(c_1 \Rightarrow_1 d_1 \wedge c_2 = d_2)$ or $(c_1 = d_1 \wedge c_2 \Rightarrow_2 d_2)$.
- The partitioning me is given as follows. For all $c = (c_1, c_2) \in \mathcal{C}$: $\mathrm{me}(c) = \{\mathcal{P}_1 \times \{c_2\} \mid \mathcal{P}_1 \in \mathrm{me}_1(c_1)\} \cup \{\{c_1\} \times \mathcal{P}_2 \mid \mathcal{P}_2 \in \mathrm{me}_2(c_2)\}$.

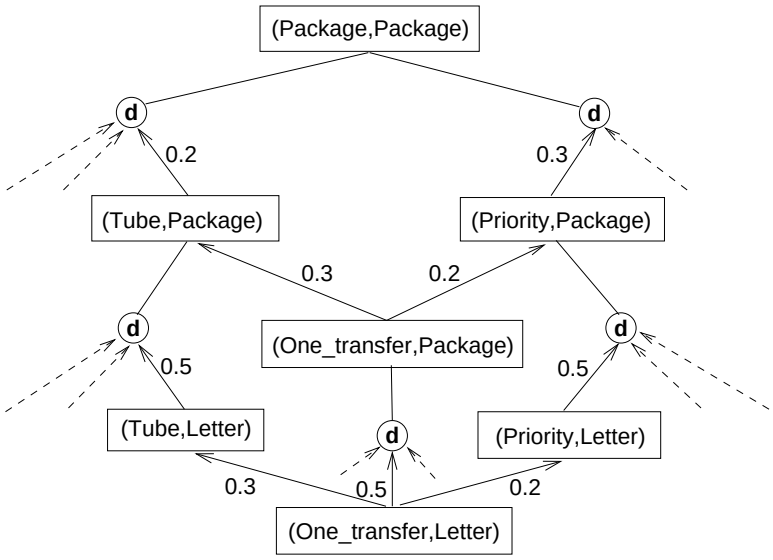

Figure 3: Natural join of schemas

- The probability assignment $\wp$ is defined as follows. For all $c = (c_1, c_2) \Rightarrow d = (d_1, d_2)$:

$$\wp(c,d) = \begin{cases} \wp_1(c_1,d_1) & \text{if } c_2 = d_2 \\ \wp_2(c_2,d_2) & \text{if } c_1 = d_1\,. \end{cases}$$

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

Example 4.30 Let $\mathbf{S}_1$ and $\mathbf{S}_2$ be the TPOB-schemas from Examples 3.6 and 4.25, respectively. Then, $\mathbf{S}_1 \bowtie \mathbf{S}_2$ is the TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ partially shown in Table 14 and Figure 3.

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]
(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, Time: [[time]], City: string, Arrive: [[time]], Shipment: [[time]]]

To define the natural join of TPOB-instances, we first need some preliminary definitions. These include the concept of intersection of two explicit values v_1 and v_2 of the same type τ, and the natural join of two explicit values v_1 and v_2 of two probabilistic tuple types τ_1 and τ_2, respectively.

Definition 4.31 (intersection of explicit values) Let v_1 and v_2 be either two values of the same classical type τ, or two explicit values of the same probabilistic type τ. 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 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 4.32 Let time be the standard calendar with respect to the linear temporal hierarchy $hour \sqsupseteq minute$, and let $\otimes$ be a conjunction strategy.

- Consider the values $v_1 = v_2 = (12, 30)$ and $v_3 = (12, 40)$ of the temporal atomic type time. Then, $v_1 \cap_{\otimes} v_2 = v_1$, while $v_1 \cap_{\otimes} v_3$ is undefined.
- Consider the following explicit values of the atomic 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 now come to our second preliminary definition — that of a natural join of two explicit values.

Definition 4.33 (natural join of explicit values) Let v_1 and v_2 be explicit values 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.

We are now ready to define the natural join of two TPOB-instances.

Definition 4.34 (natural join of TPOB-instances) Let $\mathbf{I}_1 = (\pi_1, \nu_1)$ and $\mathbf{I}_2 = (\pi_2, \nu_2)$ be two TPOB-instances over the natural-join-compatible TPOB-schemas $\mathbf{S}_1 = (\mathcal{C}_1, \sigma_1, \Rightarrow_1, \mathrm{me}_1, \wp_1)$ and $\mathbf{S}_2 = (\mathcal{C}_2, \sigma_2, \Rightarrow_2, \mathrm{me}_2, \wp_2)$, respectively. 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 defined as the TPOB-instance $\mathbf{I} = (\pi, \nu)$ over the TPOB-schema $\mathbf{S} = \mathbf{S}_1 \bowtie \mathbf{S}_2$, 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 4.35 Let $\mathbf{S}_1$ and $\mathbf{S}_2$ be the TPOB-schemas given in Example 3.6 and produced in Example 4.25, respectively. Let $\mathbf{I}_1$ and $\mathbf{I}_2$ be the TPOB-instances over $\mathbf{S}_1$ and $\mathbf{S}_2$ produced in Examples 4.11 and 4.27, respectively. Then, $\mathbf{I}_1 \bowtie_{\otimes_{ig}} \mathbf{I}_2$ is the TPOB-instance $\mathbf{I} = (\pi, \nu)$ over $\mathbf{S}_1 \bowtie \mathbf{S}_2$, where π is given by $\pi((\mathsf{Priority}, \mathsf{Priority})) = \{(o_3, o_3)\}$ and $\pi(c) = \emptyset$ for all other classes 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: $\{((8,00),[.45,1]),((8,10),[.4,1])\}$]

4.7 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 $\times$. The following condition describes when two TPOB-schemas $\mathbf{S}_1$ and $\mathbf{S}_2$ can be combined using Cartesian product.

Definition 4.36 (Cartesian-product-compatible TPOB-schemas) The TPOB-schemas $\mathbf{S}_1 = (\mathcal{C}_1, \sigma_1, \Rightarrow_1, \mathrm{me}_1, \wp_1)$ and $\mathbf{S}_2 = (\mathcal{C}_2, \sigma_2, \Rightarrow_2, \mathrm{me}_2, \wp_2)$ are *Cartesian-product-compatible* iff for all classes $c_1 \in \mathcal{C}_1$ and $c_2 \in \mathcal{C}_2$, the types $\sigma_1(c_1)$ and $\sigma_2(c_2)$ 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 $\mathbf{I}_1 \bowtie_\phi \mathbf{I}_2 = \sigma_\phi(\mathbf{I}_1 \times \mathbf{I}_2)$ over the TPOB-schema $\mathbf{S}_1 \times \mathbf{S}_2$.

Example 4.37 Let $\mathbf{S}_1$ and $\mathbf{I}_1$ be the TPOB-schema and the TPOB-instance, respectively, produced in Example 4.23. 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 and Time with Origin1 and Time1, respectively. The Cartesian product of $\mathbf{S}_1$ and $\mathbf{S}_2$ is the TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ partially shown in Table 16 and Figure 3. The conditional 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 $\mathbf{I} = (\pi, \nu)$ over the TPOB-schema $\mathbf{S}_1 \times \mathbf{S}_2$, where π is given by $\pi((\mathsf{Priority}, \mathsf{Two\text{-}transfer})) = \{(o_3, o_5)\}$ and $\pi(c) = \emptyset$ for all other classes 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, Origin1: string]
(Priority, Package)	[Origin: string, Time: ⟦time⟧, Origin1: string]
(One-transfer, Letter)	[Origin: 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])}]

4.8 Intersection, Union, and Difference

We finally define the operations of intersection, union, and difference on two TPOB-instances over the same TPOB-schema. We first describe intersection. Informally speaking, this operation intersects the sets of oids of two TPOB-instances, as well as the explicit values associated with each oid in both TPOB-instances.

Definition 4.38 (intersection of 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 $\otimes$ be a conjunction strategy. The *intersection* of $\mathbf{I}_1$ and $\mathbf{I}_2$ under $\otimes$, denoted $\mathbf{I}_1 \cap_\otimes \mathbf{I}_2$, is the 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 $o \in \pi(\mathcal{C})$.

Example 4.39 Let $\mathbf{S}$ be the TPOB-schema of Example 3.6. Let $\mathbf{I}_1$ and $\mathbf{I}_2$ be the TPOB-instances over $\mathbf{S}$ given in Example 3.8 and produced in Example 4.14, respectively. The intersection of $\mathbf{I}_1$ and $\mathbf{I}_2$ under the conjunction strategy $\otimes_{ig}$ is the TPOB-instance $\mathbf{I} = (\pi, \nu)$ over $\mathbf{S}$, where π is given by $\pi(\mathsf{Priority}) = \{o_3\}$ and $\pi(c) = \emptyset$ for all other classes 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])}]

Likewise, the union operation intuitively computes the union of the sets of oids of two TPOB-instances, combined with the union of the two explicit values associated with each oid in both TPOB-instances. We first define the union of two explicit values of the same type.

Definition 4.40 (union of explicit values) Let v_1 and v_2 be either two values of the same classical type τ, or two explicit values of the same probabilistic type τ, and let $\oplus$ be a disjunction 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

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

  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.

We are now ready to define the union of two TPOB-instances.

Definition 4.41 (union of 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 $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$. 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 $c \in \mathcal{C}$.
- $\nu(o) = \begin{cases} \nu_1(o) & \text{if } o \in \pi_1(c) - \pi_2(c) \\ \nu_2(o) & \text{if } o \in \pi_2(c) - \pi_1(c) \\ \nu_1(o) \cup_\oplus \nu_2(o) & \text{if } o \in \pi_1(c) \cap \pi_2(c). \end{cases}$

We finally define the difference of two TPOB-instances.

Definition 4.42 (difference of explicit values) Let v_1 and v_2 be either two values of the same classical type τ, or two explicit values of the same probabilistic type τ, and let $\ominus$ be a difference strategy. The *difference* of v_1 and v_2 under $\ominus$, 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.

Definition 4.43 (difference of 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 $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$. 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 $c \in \mathcal{C}$.
- $\nu(o) = \begin{cases} \nu_1(o) & \text{if } o \in \pi_1(c) - \pi_2(c) \\ \nu_1(o) -_\ominus \nu_2(o) & \text{if } o \in \pi_1(c) \cap \pi_2(c). \end{cases}$

5 The Implicit Algebra

The explicit algebra described in the preceding section suffers from many problems. First, the sizes of TPOB-instances can be very large. As we can see from Table 7, a probability must be associated with each time point involved. However, to merely say that a given package will arrive at St. Louis sometime between 5:30pm and 6:30 pm may (if we reason at a minute by minute level) require 60 time points to be specified (Table 7 only shows a couple of time points). Second, because of the large size of the explicit TPOB-instances, the costs of executing the operations is also potentially high as their inputs are large.

In this section, we alleviate this problem by defining TPOB algebraic operations on implicit TPOB-instances. These implicit operations correctly implement their explicit counterparts defined in Section 4.

5.1 Selection

In order to define the selection operation for 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 4.1.

Definition 5.1 (valuation of 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 as follows:

- 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.
- $v.P$ is undefined otherwise.

Definition 5.2 (valuation of atomic selection conditions) 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. The *probabilistic valuation* with respect to $\mathbf{I}$ and o, denoted $\mathrm{prob}_{\mathbf{I},o}$, is the following partial mapping from the set of all atomic selection conditions to the set of all closed subintervals of $[0, 1]$:

- $\mathrm{prob}_{\mathbf{I},o}(in(c)) = [\min(\mathrm{ext(c)}(o)), \max(\mathrm{ext(c)}(o))]$.
- 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 result shows that the selection on implicit TPOB-instances correctly implements its counterpart on explicit TPOB-instances. That is, the mapping ε commutes with σ_ϕ.

Theorem 5.3 (correctness of selection) *Let $\mathbf{I} = (\pi, \nu)$ be an implicit TPOB-instance over a TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$, and let ϕ be a probabilistic selection condition. Then,*

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

5.2 Restricted Selection

In order to define the restricted selection operation on implicit TPOB-instances, it is sufficient to define restricted selection on implicit values. The restricted selection on implicit TPOB-instances is then defined in the same way as in Section 4.2.

Definition 5.4 (restricted selection on implicit values) Let τ be a probabilistic tuple type. Let ϕ be of the form $P.C$, where P is a path expression for τ, and C is a constraint. Let v be an implicit value of τ. The *restricted selection on v with respect to ϕ*, denoted $\sigma^r_\phi(v)$, is defined by:

- If $v = [A_1 \colon v_1, \ldots, A_i \colon v_i, \ldots, A_k \colon v_k]$, v_i is a value of a classical type, $P = A_i$, and $v_i \in \mathrm{sol}(C)$, then $\sigma^r_\phi(v) = v$.
- If $v = [A_1 \colon v_1, \ldots, A_i \colon v_i, \ldots, A_k \colon v_k]$, v_i is an implicit value of an atomic probabilistic type, and $P = A_i$, then $\sigma^r_\phi(v) = [A_1 \colon v_1, \ldots, A_i \colon {v_i}', \ldots, A_k \colon v_k]$, where

$$v_i' \quad = \quad \{(C \wedge C', D, l, u, \delta) \mid (C', D, l, u, \delta) \in v_i,\, \mathrm{sol}(C \wedge C') \neq \emptyset\}\,.$$

- If $v = [A_1 \colon v_1, \ldots, A_i \colon v_i, \ldots, A_k \colon v_k]$, v_i is an implicit value of a probabilistic tuple type, and $P = A_i.R$, then $\sigma^r_\phi(v) = [A_1 \colon v_1, \ldots, A_i \colon \sigma^r_{R.C}(v_i), \ldots, A_k \colon v_k]$.
- Otherwise, $\sigma^r_\phi(v)$ is undefined.

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

Theorem 5.5 (correctness of restricted selection) *Let $\mathbf{I}$ be an implicit TPOB-instance over a TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$. Let ϕ be an expression of the form $P.C$, where P is a path expression for all $\sigma(c)$ with $c \in \mathcal{C}$, and C is a constraint. Then,*

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

5.3 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 4.3.

Definition 5.6 (renaming on constraints) 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 $\delta_N(v_i)$.

Definition 5.7 (renaming on implicit values) 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 5.8 (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,*

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

5.4 Natural Join

In order to define the natural join operation on implicit TPOB-instances, we need to define the intersection of implicit values. The join of implicit values and the join of TPOB-instances are then defined in the same way as in Section 4.6.

Definition 5.9 (intersection of implicit values) 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:

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

- 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 next result shows that the join on implicit TPOB-instances correctly implements its counterpart on explicit TPOB-instances. That is, the mapping ε commutes with $\bowtie_\otimes$.

Theorem 5.10 (correctness of natural join) *Let* $\mathbf{I}_1$ *and* $\mathbf{I}_2$ *be 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,*

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

5.5 Intersection, Union, and Difference

To define intersection, union, and difference, we need to define the intersection, union, and difference of implicit values, which are then extended to implicit TPOB-instances in the same way as in Section 4.8. The intersection of implicit values is given by Definition 5.9, while the union and difference of implicit values is defined below.

Definition 5.11 (union of implicit values) 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 $\oplus$ be a disjunction 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

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

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.

Definition 5.12 (difference of implicit values) 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 $\ominus$ be a difference strategy. 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

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

  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 5.13 (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,*

$$\varepsilon(\mathbf{I}_1) \cap_\otimes \varepsilon(\mathbf{I}_2) \quad = \quad \varepsilon(\mathbf{I}_1 \cap_\otimes \mathbf{I}_2)\,, \tag{1}$$

$$\varepsilon(\mathbf{I}_1) \cup_\oplus \varepsilon(\mathbf{I}_2) \quad = \quad \varepsilon(\mathbf{I}_1 \cup_\oplus \mathbf{I}_2)\,, \tag{2}$$

$$\varepsilon(\mathbf{I}_1) -_\ominus \varepsilon(\mathbf{I}_2) \quad = \quad \varepsilon(\mathbf{I}_1 -_\ominus \mathbf{I}_2)\,. \tag{3}$$

5.6 Projection, Extraction, Cartesian Product, and Conditional Join

The operations of projection, extraction, Cartesian product, and conditional join for implicit TPOB-instances are defined in exactly the same way as their counterparts for explicit TPOB-instances.

5.7 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 [2], we now define compression functions through which such implicit TPOB-instances can be made more compact.

Definition 5.14 (compression function) Let τ be an atomic probabilistic type. A *compression function* Γ for 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 5.15 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.

Definition 5.16 (compression of implicit values) 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.

Definition 5.17 (compression of implicit TPOB-instances) 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 $o \in \pi(\mathcal{C})$.

5.8 Preservation of Consistency and Coherence

We now show that all our explicit algebraic operators defined in Section 4 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). This also shows that all our implicit algebraic operators given in Section 5 preserve consistency and coherence of schemas and instances, respectively, as they correctly implement their explicit counterparts.

The explicit operators of selection, restricted selection, intersection, union, and difference trivially preserve consistency of schemas, as the input TPOB-schemas coincide with the output TPOB-schemas. Projection and renaming 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 5.18 *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 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 now concentrate on the preservation of coherence. Recall that the coherence of a TPOB-instance $\mathbf{I} = (\pi, \nu)$ over a TPOB-schema $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$ depends on π, $\mathcal{C}$, $\Rightarrow$, me, and $\wp$. The explicit algebraic 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, projection and renaming 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. Moreover, it shows that extraction preserves coherence of instances, when we do not remove any characteristic classes.

Theorem 5.19 *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. Let $\mathbf{I} = (\pi, \nu)$ and $\mathbf{S} = (\mathcal{C}, \sigma, \Rightarrow, \mathrm{me}, \wp)$. Let $\mathbf{C} \subseteq \mathcal{C}$ such that $\{c \in \mathcal{C} \mid c$ is characteristic for $\mathrm{ext}(d)(o)$ for some $d \in \mathbf{C}$ and some $o \in \pi(\mathbf{C})\} \subseteq \mathbf{C}$. Let $\mathbf{S}_1$ and $\mathbf{S}_2$ be 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.

6 Related Work

There is quite extensive work in the literature on temporal databases and temporal object-oriented databases; we refer especially to the recent surveys [19, 10] and the books [23, 22].

Probabilistic extensions to relational databases are also well-explored in the literature; see especially [16, 7] 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. [7] presented an approach that adds probabilistic uncertainty to complex value relational databases, while Kornatzky and Shimony [11, 12] and Eiter et al. [6] described approaches to probabilistic object-oriented databases. Our approach in this paper is a temporal extension of the model by Eiter et al. [6]. Additionally, the present paper newly introduces an implicit data model and an implicit algebra, which is shown to correctly implement its explicit counterpart, and which can be more efficiently realized. Moreover, the two operations of restricted selection and extraction are newly introduced here.

Even though the areas of temporal and probabilistic databases are both well-explored, there is very little work on the integration of temporal reasoning and probabilistic databases. In particular, Dyreson and Snodgrass in their pioneering work [5] and subsequently Dekhtyar et al. [2] presented approaches to temporal indeterminacy in relational databases based on probabilistic uncertainty:

- Dyreson and Snodgrass [5] 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 the *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. Our work in this paper differs from theirs 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.
- Dekhtyar et al. [2] 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. Our work in this paper, especially 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 Dekhtyar et al.'s work. Our work, however, is an extension of the much richer object-oriented data model and algebra, as compared to the relational algebra. Our work may be viewed as a generalization of theirs.

To our knowledge, there has been no work to date on temporal probabilistic object-oriented databases.

There is other work on nonprobabilistic temporal indeterminacy in databases, which is less related to our work. In particular, Snodgrass [21] models indeterminacy using a model that is based on a three-valued logic. Dutta [4] and Dubois and Prade [3] propose a fuzzy logic approach to temporal indeterminacy, while Koubarakis [14, 13] and Brusoni et al. [1] suggest approaches based on constraints. Gadia et al. [8] introduce partial temporal databases, which are based on partial temporal elements.

7 Conclusions

Dyreson and Snodgrass [5], followed subsequently by Dekhtyar et al. [2], have argued persuasively that there are numerous real-world applications where temporal uncertainty abounds. In this paper, we have used a simple example tracking shipments carried, for instance, by commercial carriers. Many other examples abound: stock market models making predictions of stock prices involve temporal probabilities specifying when a stock will reach a specific price. Archaeological databases containing radio-carbon dating of historical artifacts invariably involve temporal uncertainty as well. Programs tracking the behavior of parts on a factory floor and predicting when they will need to be serviced and/or replaced also involve temporal uncertainty. The fact that many of these applications also involve object models should come as no surprise. Descriptions of three dimensional historical artifacts are often stored using object models. Maintaining information about machine parts often includes design information, drawings, and manuals that are often represented with object models as well.

In this paper, we have made a first attempt to deal with temporal uncertainty in object-based systems. We have provided two models. The first is an explicit model where a probability is associated with each time point. As temporal granularity gets finer and finer, this model gets more and more impractical to use. For this explicit model, we provide an algebra (e-algebra) that extends the relational algebra.

To avoid the problems associated with the e-algebra, we introduce a succinct implicit algebra (i-algebra). We define operators for the i-algebra. We show that each operator in the i-algebra correctly implements the corresponding operator in the e-algebra without computing the entire explicit representation. Thus, the e-algebra operators "work" on a compact implicit represent of a much larger explicit representation.

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 supported by the Army Research Lab under contract number DAAL-0197K0135, the Army Research Office under grant number DAAD190010484, by DARPA/RL contract number F306029910552, by the ARL CTA on Advanced Decision Architectures, by a DFG grant, and by a Marie Curie Individual Fellowship of the European Community.

Appendix A. Proofs for Section 5

For the proof of Theorem 5.3, 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 7.1 *Let P be a path expression for the probabilistic type τ, and v be an implicit value of τ. Then,*

$$\varepsilon(v.P) \quad = \quad \varepsilon(v).P\,. \tag{4}$$

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,

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

Proof of Theorem 5.3. 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 7.1 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 7.1 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 5.5. 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 5.8. 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 $\varepsilon(\delta_N(v)) = \delta_N(\varepsilon(v))$. $\square$

Proof of Theorem 5.10. 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,

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

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 $\varepsilon(v_1 \cap_\otimes v_2) = \varepsilon(v_1) \cap_\otimes \varepsilon(v_2)$. $\square$

Proof of Theorem 5.13. Equation (1) follows immediately from the proof of Theorem 5.10. We next prove Equation (2). 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,

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

We finally prove Equation (3). 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 B. Proofs for Section 5.8

Proof of Theorem 5.18. (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 5.19. (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. □

References

[1] 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.

[2] 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`.

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

[4] 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.

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

[6] T. Eiter, J. J. Lu, T. Lukasiewicz, and V. S. Subrahmanian. Probabilistic object bases. *ACM Transactions on Database Systems*, September 2001. To appear.

[7] T. Eiter, T. Lukasiewicz, and M. Walter. A data model and algebra for probabilistic complex values. *Annals of Mathematics and Artificial Intelligence*. To appear.

[8] 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.

[9] 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.

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

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

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

[13] 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.

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

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

[16] 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.

[17] 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.

[18] 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.

[19] 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.

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

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

[22] 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.

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