Probabilistic Temporal Databases, I: Algebra

Alex Dekhtyar, dekhtyar@cs.uky.edu
Dept. of Computer Science, University of Kentucky, Lexington, KY, 40507,
and
Robert Ross and V. S. Subrahmanian, {robross,vs}@cs.umd.edu
Dept. of Computer Science, Institute for Advanced Computer Studies and Institute for Systems Research, University of Maryland, College Park, MD 20742,

---

Dyreson and Snodgrass have drawn attention to the fact that in many temporal database applications, there is often uncertainty present about the start time of events, the end time of events, the duration of events, etc. When the granularity of time is small (e.g. milliseconds), a statement such as "Packet p was shipped sometime during the first 5 days of January, 1998" leads to a massive amount of uncertainty ($5 \times 24 \times 60 \times 60 \times 1000$) possibilities. As noted in [41], past attempts to deal with uncertainty in databases have been restricted to relatively small amounts of uncertainty in attributes. Dyreson and Snodgrass have taken an important first step towards solving this problem.

In this paper, we first introduce the syntax of Temporal-Probabilistic (TP) relations and then show how they can be converted to an explicit, significantly more space-consuming form called Annotated Relations. We then present a *Theoretical Annotated Temporal Algebra* (TATA). Being explicit, TATA is convenient for specifying how the algebraic operations should behave, but is impractical to use because annotated relations are overwhelmingly large.

Next, we present a *Temporal Probabilistic Algebra* (TPA). We show that our definition of the TP-Algebra provides a correct implementation of TATA despite the fact that it operates on implicit, succinct TP-relations instead of the overwhelmingly large annotated relations. Finally, we report on timings for an implementation of the TP-Algebra built on top of ODBC.

Categories and Subject Descriptors: H.2.1 [**Database Management**]: Logical Design—*Data Models*; H.2.3 [**Database Management**]: Languages—*Query Languages*

General Terms: probabilistic databases, temporal databases, relational algebra

---

A complete version of this paper containing proofs and additional experiments is available [9]. It may be obtained electronically from ftp://ftp.cs.umd.edu/pub/papers/papers/ncstrl.umcp/CS-TR-3987/CS-TR-3987.ps.Z

Different parts of this work were supported by the Army Research Office under Grants DAAH-04-96-10297, DAAG-55-97-10047 and DAAH04-96-1-0398, by the Army Research Laboratory under contract number DAAL01-97-K0135, by an NSF Young Investigator award IRI-93-57756, by a DARPA/Rome Labs contract F306029910552 and by a grant from Lockheed Martin Advanced Technology Laboratories.

This work had been completed by the first author while at the Dept. of Computer Science, University of Maryland, College Park.

1. INTRODUCTION

The world we live in evolves dynamically over time. Furthermore, our knowledge about what is true in the world at a fixed point in time is highly uncertain. Databases that attempt to capture temporal aspects of the world encounter uncertainty in a variety of applications.

—**Scheduling:** Consider the databases maintained by a transportation provider such as CSX or Federal Express. When a package is delivered to such an organization for shipping, a tentative shipping *schedule* is created for the package. The transportation provider must maintain such schedules for millions of packages. Such schedules specify which flight (or truck) the shipment is scheduled to leave on, when the shipment will reach a way point, and so on. However, there is uncertainty about how long a particular part of the schedule will actually take. For example, Federal Express may ship a package from Boston to Chicago via Albany, NY. They have reliable statistics on how long the Boston-Albany leg takes, and how long the Albany-Chicago leg takes. A user who wants to know when his shipment is likely to reach him usually gets an *uncertain* answer of the form "Either today (36%) or tomorrow (64%)." A database system used by such a transportation vendor must have the ability to handle temporal modes of uncertainty.

—**Weather Applications:** Consider a weather database that tracks the weather at a fixed location (e.g. Washington). Such a weather database contains not only information about the weather in Washington in the past, but also contains projections for the future. Needless to say, any prediction about the future is liable to be uncertain. How often have we heard a TV newscaster say "There is a 39% probability of rain this afternoon"?

—**Time-Series Stock Applications:** There are a wide variety of programs that analyze the behavior of stocks, and predict their rise and fall in the future. Most such programs associate with their predictions a level of uncertainty. Such programs may say "We expect, with 60-70% certainty, that IBM stock will fall by 30% sometime in the next 2 weeks." When the output of such programs is to be stored in a relational database system, we must have the ability to represent and manipulate such statements.

All the above applications require the ability to make statements of the following kind: *Data tuple d is in relation R at some point of time in the interval $[t_i, t_j]$ with probability between p and p'.* For example, in the Transportation Application above, we should be able to store statements of the form "Package p will arrive in Albany at some time between 9am and 5pm on Nov. 8 with probability 50–60%." Similarly, in the weather application, we should be able to store statements of the form "Rain is expected to begin sometime between 2pm and 12 midnight on Nov. 8 with probability 5–20%." In the stock market application, we should be able to store statements of the form "IBM stock will reach $300 per share some time during the time interval Nov 1-10 with probability 90-100%." The main contributions of this paper may now be summarized as follows.

• First, in Section 3 we introduce the concept of a *temporal-probabilistic tuple* or TP-tuple, for short. Intuitively, a TP-tuple allows us to augment classical relational database tuples with temporal-probabilistic data, as well as arbitrary probability distributions. For example, not only can we say "Data tuple d is in relation r at some point of time in the interval $[t_i, t_j]$ with probability between p and p'" but we can also say that the probability mass is distributed over $[t_i, t_j]$ according to an *arbitrary* probability distribution. Throughout this paper, we will introduce definitions which allow us to make such statements in a TP-relation and which allow us to manipulate such TP-relations algebraically.

• Then, in Section 4 we show how given any TP-tuple tp, we may "flatten" tp into a set of *annotated tuples*. In general, the set of annotated tuples associated with a single TP-tuple can be very large — hence, annotated tuples serve as a *purely theoretical device*.

• We then define a *Theoretical Annotated Temporal Algebra* (TATA) in Section 5 and show how the classical relational algebra operations can be extended to the case of annotated tuples. Intuitively, the Theoretical Annotated Temporal Algebra provides a theoretical specification of how the TP-Algebra operations must be defined.

• We proceed in Section 6 by defining a *Temporal-Probabilistic Algebra* (TPA) which directly manipulates TP-tuples *without converting them to annotated tuples.* This has a great advantage, as TP-tuples are very succinct objects. We show that for each operation α in the Theoretical Annotated Temporal Algebra, there is a corresponding operation α' in the Temporal-Probabilistic Algebra which precisely captures it. Thus, the Temporal-Probabilistic Algebra is a sound and complete way of implementing the declarative semantics for temporal probabilistic data prescribed by the Theoretical Annotated Temporal Algebra. The correctness results are formally proved for every operation.

• In Sections 5 and 6 we also show how each operator, whether in the TP-Algebra, or in the Theoretical Annotated Temporal Algebra, can be parameterized by the user's knowledge of the dependencies between events. This is important because, as shown in [23], the probability of a complex event like $(e_1 \vee e_2)$ depends upon our knowledge of the dependencies between e_1 and e_2.

• Finally, in Section 7 we present an implementation of the TP-Algebra on top of ODBC and provide a set of experimental results.

2. PRELIMINARIES AND BASIC DEFINITIONS

In this section, we provide some basic definitions that are used in the algebras we develop later in the paper. The work reported in sections 2.1, 2.2, 2.3 and 2.4 is not new, but forms the basic definitions needed to describe our algebras. Section 2.5 describes new work.

Section 2.1 contains the description of a notion of a *calendar*, borrowing from definitions in [22]. Calendars are needed because all TP-relations will assume that time is specified w.r.t. an arbitrary but fixed calendar. Section 2.2 defines temporal constraints over an arbitrary but fixed calendar. As specified earlier in the paper, our algebras use constraints to describe sets of time points. Section 2.3 describes *distribution functions*. Section 2.4 specifies a set of axioms that a function must satisfy for it to be considered a probabilistic conjunction or disjunction strategy. As it is known that the probability of conjunctive and disjunctive combinations of two or more events depends not only on the probabilities of the events themselves but also on the dependencies between them, there is no unique way of computing such probabilities. This section specifies what axioms a function must satisfy for it to be considered a possible way of computing the probabilities of compound events given the probabilities of simple events. Section 2.5, finally, contains a description of how conflicting information about the probability of an event (which must be true at a certain time point) can be combined together. We introduce the concept of a *combination function* as a function that combines a set of probability intervals into one interval while satisfying a prerequisite set of axioms.

2.1 Calendars

In this section, we define the concept of a *calendar* that is used by a TP application. In our architecture, a TP application assumes the existence of an arbitrary but fixed calendar. The definitions in this section are not new, but taken from [22].

DEFINITION 2.1 (TIME UNIT). A *time unit* consists of a *name* and a *time-value set*. The time-value set has a linear order, denoted $<_T$, where T is the name of the time unit. As usual, we let $\leq_T$ denote the reflexive closure of the $<_T$ relation. A time unit is either *finite* or *infinite*, depending on whether its time-value set is finite or infinite; an infinite time-value set is assumed to be countable. □

For instance, the time units named *day*, *month*, and *year* may have the time-value sets $\{1, \ldots, 31\}$, $\{1, \ldots, 12\}$, and $\{$all integers$\}$ respectively.

DEFINITION 2.2 (LINEAR HIERARCHY). A *linear hierarchy of time units*, denoted H, is a finite collection of distinct time units with a linear order $\sqsubseteq$ among those time units. The greatest time unit according to $\sqsubseteq$ may be either finite or infinite, while all other time units in the hierarchy must be finite. □

For instance, $H_1 = day \sqsubseteq month \sqsubseteq year$, $H_2 = minute \sqsubseteq hour \sqsubseteq day \sqsubseteq month \sqsubseteq year$, and $H_3 = hour \sqsubseteq day \sqsubseteq month$ are all linear hierarchies of time units.

DEFINITION 2.3 (TIME POINT). Suppose $T_1 \sqsubseteq \cdots \sqsubseteq T_n$ is a linear hierarchy H of time units. A *time point* t in H is an n-tuple $(v_1, \ldots, v_n)$ such that for all $1 \leq i \leq n$, v_i is a time-value in the time-value set of T_i. Let $(v_1/.../v_n)$ be an abbreviation for time point t.

Time points are ordered according to the lexicographic ordering $<_H$ which is defined in the usual way. Thus, time point $t = (v_1, \ldots, v_n) <_H t' = (v'_1, \ldots, v'_n)$ iff there exists an i $(1 \leq i \leq n)$ such that $v_i <_{T_i} v'_i$ and $v_j = v'_j$ for all $j = i+1, \ldots, n$. Note that if $i = n$, then the $(v_j = v'_j)$ statement is vacuously true. When $t <_H t'$, we say that t *occurs before* t', and conversely, t' *occurs after* t. If $t = t'$, we say that t *occurs simultaneously with* t'. □

A time point in linear hierarchy H is simply an instantiation of each time unit in H (a specific point in time with respect to H). For instance, using hierarchy H_1 given above, "March 16, 1997" could be specified by the time point (16/3/1997). By using hierarchy H_2 given above, "3:45pm on March 12, 1997" could be specified by the time point (45, 15, 12, 3, 1997). For hierarchy H_1 given above, time point t *occurs before or simultaneously with* t', denoted $t = (v_{day}, v_{month}, v_{year}) \leq_{H_1} t' = (v'_{day}, v'_{month}, v'_{year})$, is true iff $((v_{year} < v'_{year}) \vee (v_{year} = v'_{year} \wedge v_{month} < v'_{month}) \vee (v_{year} = v'_{year} \wedge v_{month} = v'_{month} \wedge v_{day} \leq v'_{day}))$.

DEFINITION 2.4 (CALENDAR). A *calendar* τ consists of a linear hierarchy H of time units and a *validity predicate* denoted $valid_H$ (or simply *valid* if H is clear from context). A validity predicate specifies a non-empty set of valid time points; $valid_H(t)$ is true iff t is a valid time point. The *set of all time points over calendar* τ, denoted S_τ, is defined as $\{t \mid t$ is a time point in H and $valid_H(t)$ is true$\}$. □

For instance, if we are representing the Gregorian calendar τ by hierarchy H_1 given above, a suitable validity predicate states that valid(14/3/1996) = true but valid(29/2/1997) = false. (29/2/1997) is not a valid time point since February of 1997 only contains 28 days. Note that a calendar for the hierarchy $dayOfWeek \sqsubseteq day \sqsubseteq month \sqsubseteq year$ should have only one valid time point for each instantiation of (*day*, *month*, *year*) since these three time units uniquely determine the valid time-value for *dayOfWeek*.

Let $next_\tau(t)$ denote the next, consecutive time point after t. Thus, $next_\tau(t)$ denotes the time point $t' \in S_\tau$ where t' occurs after t and for all other $t'' \in S_\tau$ where t'' occurs after t, t'' also occurs after t'.

2.2 Constraints

When expressing a statement of the form "Data tuple d is in relation r at some time point in a set T of time points with probability in the interval $[p_1, p_2]$ and with the probability distributed according to distribution δ", we must be able to specify the set T of time points. Constraints are a natural way of specifying such sets. In this section, we recapitulate (from [22]) how *temporal constraints* can be used to specify sets of time points associated with a calendar.

DEFINITION 2.5 (ATOMIC TEMPORAL CONSTRAINT). Suppose $T_1 \sqsubseteq \cdots \sqsubseteq T_n$ is a linear hierarchy H of time units over calendar τ. An *atomic temporal constraint over calendar* τ must take one of the following forms:
1. $(T_i\ op\ v_i)$ where *op* is a member of the set $\{\leq, <, =, \neq, >, \geq\}$ and v_i is a time-value in the time-value set of time unit T_i. Here, $(T_i\ op\ v_i)$ is called an *atomic time-value constraint.*
2. $(t_1 \sim t_2)$ where $t_1, t_2 \in S_\tau$ and $t_1 \leq_H t_2$. Here, $(t_1 \sim t_2)$ is called an *atomic time-interval constraint.* For convenience, let (t_1) be an abbreviation for $(t_1 \sim t_1)$.
□

For example, $(day < 15)$, $(month \geq 8)$, and (12/3/1997 ∼ 10/4/1997) are all atomic temporal constraints, but $(1996 = year)$ is not. Also, $(day < 45)$ is not an atomic temporal constraint since 45 is not in the time-value set of *day*. Similarly, (15/2/1997 ∼ 29/2/1997) is not an atomic temporal constraint since (29/2/1997) is not a valid time point in τ. Furthermore, (10/4/1997 ∼ 12/3/1997) is not an atomic temporal constraint since time point (10/4/1997) *occurs after* (12/3/1997).

DEFINITION 2.6 (TEMPORAL CONSTRAINT). A *temporal constraint C over calendar τ* is defined inductively in the following way:
- Any atomic temporal constraint over τ is a temporal constraint over τ.
- If C_1 and C_2 are temporal constraints over τ, then $(C_1 \wedge C_2)$, $(C_1 \vee C_2)$, and $(\neg C_1)$ are temporal
constraints over τ.

If temporal constraint C is solely a boolean combination of *atomic time-value* *constraints*, then C is a *time-value constraint*. Similarly, if temporal constraint C is solely a boolean combination of *atomic time-interval constraints*, then C is a *time-interval constraint*. □

For instance, $((day > 5 \wedge day < 15) \wedge (month = 4 \vee month \geq 8) \wedge year = 1996)$ and $((12/3/1997 \sim 10/4/1997) \vee (10/7/1997 \sim 10/7/1997))$ are temporal constraints but $(day > 5 \neg \wedge\ day < 15)$ is not.

Definition 2.7 (solution set to a temporal constraint). Suppose $T_1 \sqsubseteq \cdots \sqsubseteq T_n$ is a linear hierarchy H of time units over calendar τ. Then an atomic temporal constraint over τ is of the form $(T_i\ op\ v_i)$ or $(t_1 \sim t_2)$. The *solution set to an atomic temporal constraint over calendar* τ is is defined in the table below.

Case	S
$op = (\leq)$	$S = \{t \mid t \in S_\tau \wedge t.T_i \leq_{T_i} v_i\}$
$op = (<)$	$S = \{t \mid t \in S_\tau \wedge t.T_i <_{T_i} v_i\}$
$op = (=)$	$S = \{t \mid t \in S_\tau \wedge t.T_i = v_i\}$
$op = (\neq)$	$S = \{t \mid t \in S_\tau \wedge t.T_i \neq v_i\}$
$op = (>)$	$S = \{t \mid t \in S_\tau \wedge v_i <_{T_i} t.T_i\}$
$op = (\geq)$	$S = \{t \mid t \in S_\tau \wedge v_i \leq_{T_i} t.T_i\}$
$op = (\sim)$	$S = \{t \mid t \in S_\tau \wedge t_1 \leq_H t \leq_H t_2\}$

The *solution set* $sol(C)$ *to a non-atomic temporal constraint* C *over calendar* τ, is defined inductively as shown in the following table:

Case	S
C is an atomic temporal constraint	$S = \mathrm{sol}(C)$
$C = (C_1 \wedge C_2)$	$S = \mathrm{sol}(C_1) \cap \mathrm{sol}(C_2)$
$C = (C_1 \vee C_2)$	$S = \mathrm{sol}(C_1) \cup \mathrm{sol}(C_2)$
$C = \neg C_1)$	$S = S_\tau - sol(C_1)$

Each time point $t \in \mathrm{sol}(C)$ is called *a solution to* C. □

For instance, the solution set to $(day > 25)$ over the Gregorian calendar τ is the set of all time points $(day,\ month,\ year) \in \tau$ where $day > 25$. Note that (29, 2, 1996) is in this set but (29, 2, 1997) is not since the latter is not a valid time point in S_τ. Also, the solution set to $(1/1/1996 \sim 31/12/1996)$ over the Gregorian calendar would contain 366 time points (one for each calendar day in 1996) while the solution set to $(1/1/1997 \sim 31/12/1997)$ over the same calendar would contain 365 time points.

For example, the solution set to $((5/8/1997 \sim 10/8/1997) \vee (7/8/1997 \sim 12/8/1997))$ would contain eight time points.

The following well known result states that any time-value constraint can be rewritten as an equivalent time-interval constraint (i.e., one which has an equal solution set) and vice-versa.

Proposition 1 (Folk theorem). *Time-value constraints and time-interval constraints have the same expressive power.*

Calendar τ is a *finite calendar* iff S_τ is finite. When all time units in a calendar are finite, the calendar itself is also finite. Furthermore, for all temporal constraints

C over a finite calendar τ, $\mathrm{sol}(C)$ must be a finite subset of S_τ. Given a finite calendar τ, we use t_S^τ to denote the smallest time point of τ (w.r.t. the ordering $<_H$ associated with the calendar) and t_E^τ to denote the largest time point. When τ is clear from context, we will drop the superscript τ and just write t_S and t_E.

In the rest of this paper, all calendars are assumed to be finite unless we specifically state otherwise. Furthermore, all of our examples will use a finite version of the Gregorian calendar.

2.3 Distribution functions

Consider a simple statement saying that data tuple d is in relation r at some time point in the set $\{1, 2, 3, 4\}$ with probability 0.7. Suppose we are now asked "what the probability that d is in r at time 2?" There is no way to answer this question without assuming the existence of some probability distribution. In this paper, we wish to allow designers of TP-databases to specify probability distributions for each set of time points.

Definition 2.8 (probability distribution function). Let D be a temporal constraint over calendar τ such that $|\mathrm{sol}(D)| \geq 1$. Then a ***probability distribution function (PDF) over calendar*** τ, denoted $\mathrm{pdf}(D, t_j)$, is a function which takes D and a time point $t_j \in S_\tau$ as input, and returns as output a probability ρ_j which satisfies the following conditions:
1. For each $t_j \in S_\tau$, $0 \leq \rho_j \leq 1$.
2. For all $t_j \in S_\tau$ where $t_j \notin \mathrm{sol}(D)$, $\rho_j = 0$.
3. $\sum_{t_j \in S_\tau} \rho_j \leq 1$. This implies that $\sum_{t_j \in \mathrm{sol}(D)} \mathrm{pdf}(D, t_j) \leq 1$.

If the sum $\sum_{t_j \in S_\tau}(\rho_j)$ is strictly less than one, the function is called a *partial PDF*; if the sum is exactly equal to one, the function is called a *complete PDF*. A PDF is *determinate* if $\sum_{t_j \in S_\tau}(\rho_j)$ is computable in constant time. □

PDFs are both discrete and finite. Complete PDFs tell us what percentage of the total probability mass (i.e., 1.0) is associated with each $t_j \in \mathrm{sol}(D)$. Partial PDFs are useful when modeling infinite distributions; here, we are considering only a finite portion of the total probability mass. Determinate PDFs tell us up-front that a fixed percentage of the probability mass is unassigned. Thus, every complete PDF is determinate. In addition, a partial PDF which is known to allocate only a total of 0.9 to the values in S_τ is determinate.

To see how specific PDFs may be defined, let us examine some examples.

Example 2.1 (PDF; uniform). The *PDF for the uniform distribution over calendar* τ, denoted $\mathrm{pdf}_u(D, t_j)$, is defined as $\rho_j = \frac{1}{|\mathrm{sol}(D)|}$ if $t_j \in \mathrm{sol}(D)$ or $\rho_j = 0$ otherwise. pdf_u is a complete PDF.

Notice that for all $t_1, t_2 \in \mathrm{sol}(D)$, $\rho_1 = \rho_2$. In other words, we are equally dividing the probability mass among all of the relevant time points. Also, $\sum_{t_j \in S_\tau}(\rho_j) = 1$ is clearly true since there are $n = |\mathrm{sol}(D)|$ non-zero ρ_js, one for each $t_j \in \mathrm{sol}(D)$, and $n \cdot \rho_j = |\mathrm{sol}(D)| \cdot \frac{1}{|\mathrm{sol}(D)|} = 1$. Furthermore, we will never divide by zero since by definition of PDFs, $|\mathrm{sol}(D)| \geq 1$.

For the following PDF examples, let D be a temporal constraint and let $t_0, \ldots, t_n$ be a list of distinct time points in S_τ where $\mathrm{sol}(D) = \{t_0, \ldots, t_n\}$ and t_i *occurs before* t_{i+1} for all $0 \leq i < n$, i.e. $sol(D)$ is enumerated in ascending order of time points. For instance if $D = (1/8/1997 \sim 3/8/1997)$ then $n = 2$, $t_0 = 1/8/1997$, $t_1 = 2/8/1997$, and $t_2 = 3/8/1997$.

EXAMPLE 2.2 (PDF; GEOMETRIC). Let p be a probability where $(0 < p < 1)$. Then the *PDF for the geometric distribution with parameter p over calendar τ*, denoted $\mathrm{pdf}_{g,p}(D, t_j)$, is defined as $\rho_j = p \cdot (1-p)^i$ if $t_j = t_i \in \mathrm{sol}(D)$ or $\rho_j = 0$ otherwise. pdf_g is a partial PDF. Note that if $|\mathrm{sol}(D_j)|$ is fixed (or constant time computable), then pdf_g is a determinate PDF.

If $p = \frac{1}{3}$, $\mathrm{pdf}_{g,p}(D, t_0) = \frac{1}{3} \cdot (\frac{2}{3})^0$, $\mathrm{pdf}_{g,p}(D, t_1) = \frac{1}{3} \cdot (\frac{2}{3})^1$, and $\mathrm{pdf}_{g,p}(D, t_2) = \frac{1}{3} \cdot (\frac{2}{3})^2$. Notice that if $p = \frac{1}{2}$, then $\mathrm{pdf}_{g,p}(D, t_0) = \frac{1}{2}$ and $\mathrm{pdf}_{g,p}(D, t_i)$ will be half of $\mathrm{pdf}_{g,p}(D, t_{i-1})$ for each $1 \leq i \leq n$.

Let $\mathrm{pdf}_{g_c,p}$ be defined in the same way as $\mathrm{pdf}_{g,p}$ except $\mathrm{pdf}_{g_c,p}(D, t_n) = 1$ if $|\mathrm{sol}(D)| = 1$ or $\mathrm{pdf}_{g_c,p}(D, t_n) = 1 - (\sum_{j=0}^{n-1}(\mathrm{pdf}_{g,p}(D, t_j)))$ otherwise. We call $\mathrm{pdf}_{g_c,p}$ the *complete correlate* of $\mathrm{pdf}_{g,p}$ since $\mathrm{pdf}_{g_c,p}(D, t_j) = \mathrm{pdf}_{g,p}(D, t_j)$ for all $t_j \in S_\tau - \{t_n\}$ and since $\mathrm{pdf}_{g_c,p}$ is a complete PDF. In general, one can construct a complete correlate for any partial PDF in a similar way. Note that when $p = \frac{1}{2}$ and $|\mathrm{sol}(D)| > 1$, $\mathrm{pdf}_{g_c,p}$ has the nice property that $\mathrm{pdf}_{g_c,p}(D, t_n) = \mathrm{pdf}_{g_c,p}(D, t_{n-1})$.

EXAMPLE 2.3 (PDF; BINOMIAL). Let p be a probability where $(0 < p < 1)$. Then the *PDF for the binomial distribution with parameter p over calendar τ*, denoted $\mathrm{pdf}_{b,p}(D, t_j)$, is defined as $\rho_j = \binom{n}{i} \cdot p^i \cdot (1-p)^{n-i}$ if $t_j = t_i \in \mathrm{sol}(D)$ or $\rho_j = 0$ otherwise. pdf_b is a complete PDF.

EXAMPLE 2.4 (PDF; POISSON). Let $(\lambda > 0)$ be a rate and let e be the base of the natural logarithm (i.e., $e \simeq 2.71828$). Then the *PDF for the Poisson distribution with parameter λ over calendar τ*, denoted $\mathrm{pdf}_{po,\lambda}(D, t_j)$, is defined as $\rho_j = e^{-\lambda} \cdot \frac{\lambda^i}{i!}$ if $t_j = t_i \in \mathrm{sol}(D)$ or $\rho_j = 0$ otherwise. pdf_{po} is a partial PDF. When $|\mathrm{sol}(D)|$ is known, then pdf_{po} is a determinate PDF.

Techniques that specify how to associate and store probability distributions with events are provided by Dyreson and Snodgrass [11, p. 8] and by Dey and Sarkar [10]. Hence, we do not discuss this in further detail.

Throughout the rest of this paper, we will use (δ = "u"), (δ = "g, p"), (δ = "g_c, p"), (δ = "b, p"), and (δ = "po, λ") to represent the distribution functions for pdf_u, $\mathrm{pdf}_{g,p}$, $\mathrm{pdf}_{g_c,p}$, $\mathrm{pdf}_{b,p}$, and $\mathrm{pdf}_{po,\lambda}$ respectively. Furthermore, unless we specifically state otherwise, assume that parameter $p = 0.5$. Thus, (δ = "g") represents the $\mathrm{pdf}_{g,0.5}$ function.

Temporal Probabilistic Databases studied in this paper will use pdfs to store compactly the probabilstic information.[1]

2.4 Probabilistic strategies

Given the probabilities p_1 and p_2 of events e_1 and e_2, how do we compute the probability p of compound event $(e_1 \wedge e_2)$? As argued in [23], the answer depends

[1]One key feature of pdfs is the fact that they define a known probability distribution. Situations when the user does not have information about probability distribution theoretically can be taken care of by introducing a special kind of distribution function: *ignorance distribution*. However, this would require changes in the semantics of our relations, many relational algebra operators would have to be extended with a special case handling the ignorance distribution. For the sake of clarity we will not consider such distributions in this paper.

on the relationship between e_1 and e_2. For instance if e_1 and e_2 are *mutually exclusive*, p should be zero; if e_1 and e_2 are independent of each other, p should be $(p_1 \cdot p_2)$. A similar situation arises when computing the probability of $(e_1 \vee e_2)$. We address these problems by consulting *probabilistic conjunction strategies* and *probabilistic disjunction strategies*. Both of these concepts were originally defined in [23] and are recapitulated below.

Before proceeding, recall that intervals obey the following definitions/properties:
1.$[L_1, U_1] \leq [L_2, U_2]$ iff $(L_1 \leq L_2 \wedge U_1 \leq U_2)$.
2.$[L_1, U_1] \geq [L_2, U_2]$ iff $(L_1 \geq L_2 \wedge U_1 \geq U_2)$.
3.$[L_1, U_1] \subseteq [L_2, U_2]$ iff $(L_1 \geq L_2 \wedge U_1 \leq U_2)$.
4.$[L, U] = ([L_1, U_1] \cap [L_2, U_2])$ **iff** $(L = \max(L_1, L_2) \wedge U = \min(U_1, U_2) \wedge L \leq U)$.

Definition 2.9 (probabilistic conjunction/disjunction strategy). *Let events e_1, e_2 be associated probabilistic intervals $[L_1, U_1]$ and $[L_2, U_2]$ respectively. Then a* probabilistic conjunction strategy *(*probabilistic disjunction strategy*) is a binary operation $\otimes$ ($\oplus$) which uses this information to compute the probabilistic interval $[L, U]$ for event "$e_1 \wedge e_2$" ("$e_1 \vee e_2$"). When the events involved are clear from context, we use $[L, U] = [L_1, U_1] \otimes [L_2, U_2]$ to denote $(e_1 \wedge e_2, [L, U]) = (e_1, [L_1, U_1]) \otimes (e_2, [L_2, U_2])$ and we use $[L, U] = [L_1, U_1] \oplus [L_2, U_2]$ to denote $(e_1 \vee e_2, [L, U]) = (e_1, [L_1, U_1]) \oplus (e_2, [L_2, U_2])$. Every conjunctive (disjunctive) strategy must conform to the following postulates:*

Generic postulates ($* \in \{\otimes, \oplus\}$)	
1. Commutativity	$([L_1, U_1] * [L_2, U_2]) = ([L_2, U_2] * [L_1, U_1])$
2. Associativity	$(([L_1, U_1] * [L_2, U_2]) * [L_3, U_3]) = ([L_1, U_1] * ([L_2, U_2]) * [L_3, U_3]))$
3. Monotonicity	$([L_1, U_1] * [L_2, U_2]) \leq ([L_1, U_1] * [L_3, U_3])$ *if* $[L_2, U_2] \leq [L_3, U_3]$
Probabilistic Conjunction postulates	
4.a. Bottomline	$([L_1, U_1] \otimes [L_2, U_2]) \leq [\min(L_1, L_2), \min(U_1, U_2)]$
5.a. Identity	$([L_1, U_1] \otimes [1, 1]) = [L_1, U_1]$
6.a. Annihilator	$([L_1, U_1] \otimes [0, 0]) = [0, 0]$
7.a. Ignorance	$([L_1, U_1] \otimes [L_2, U_2]) \subseteq [\max(0, L_1 + L_2 - 1), \min(U_1, U_2)]$
Probabilistic Disjunction postulates	
4.b. Bottomline	$([L_1, U_1] \oplus [L_2, U_2]) \geq [\max(L_1, L_2), \max(U_1, U_2)]$
5.b. Identity	$([L_1, U_1] \oplus [0, 0]) = [L_1, U_1]$
6.b. Annihilator	$([L_1, U_1] \oplus [1, 1]) = [1, 1]$
7.b. Ignorance	$([L_1, U_1] \oplus [L_2, U_2]) \subseteq [\max(L_1, L_2), \min(1, U_1 + U_2)]$

Postulates 1–6 follow from the well-known properties of the probabilities of conjunctions and disjunctions. A brief explanation of axiom 7 is in order. Boole proved in 1854 [4] that if events e_1, e_2 are known to have probabilities in the intervals $[L_1, U_1], [L_2, U_2]$, *and we do not know anything about the relationship between these two events*, then the best that can be said about the probability of $(e_1 \wedge e_2)$ is that it lies in the interval shown above. Similarly, $[\max(L_1, L_2), \min(1, U_1 + U_2)]$ had been established by Boole as the probabilistic interval for the disjunction of e_1 and e_2. This forms the basis for numerous pieces of work in the AI and deductive database literature ([16; 27; 29] to name a few). This axiom merely says that if we know something about the dependency between e_1, e_2, then we must be able to infer a tighter probability interval than complete ignorance about dependencies

would allow us to infer.
The following are some sample conjunctive and disjunctive strategies ([23]):

Conjunctive Strategies	
Ignorance	$([L_1, U_1] \otimes_{ig} [L_2, U_2]) = [\max(0, L_1 + L_2 - 1), \min(U_1, U_2)]$
Positive Correlation	$([L_1, U_1] \otimes_{pc} [L_2, U_2]) = [\min(L_1, L_2), \min(U_1, U_2)]$
Negative Correlation	$([L_1, U_1] \otimes_{nc} [L_2, U_2]) = [\max(0, L_1 + L_2 - 1), \max(0, U_1 + U_2 - 1)]$
Independence	$([L_1, U_1] \otimes_{in} [L_2, U_2]) = [L_1 \cdot L_2, U_1 \cdot U_2]$
Disjunctive Strategies	
Ignorance	$([L_1, U_1] \oplus_{ig} [L_2, U_2]) = [\max(L_1, L_2), \min(1, U_1 + U_2)]$
Positive Correlation	$([L_1, U_1] \oplus_{pc} [L_2, U_2]) = [\max(L_1, L_2), \max(U_1, U_2)]$
Negative Correlation	$([L_1, U_1] \oplus_{nc} [L_2, U_2]) = [\min(1, L_1 + L_2), \min(1, U_1 + U_2)]$
Independence	$([L_1, U_1] \oplus_{in} [L_2, U_2]) = [L_1 + L_2 - (L_1 \cdot L_2), U_1 + U_2 - (U_1 \cdot U_2)]$

Note that we use the more general notion of a probability interval $[L, U] \subseteq [0, 1]$ instead of a point probability $p \in [0, 1]$; intervals allow us to reason about the probabilities of compound events (through operators such as $\otimes_{ig}$) without making traditional assumptions like independence [23].

Probabilistic conjunctions will be useful when describing TATA and TPA semantics for *cartesian products* (§5.6, §6.7). As conjunctive and disjunctive probabilistic strategies are commutative and associative, we can extend the definition of either strategy to apply to more than two arguments. We adopt the notations $([L_1, U_1] \otimes \ldots \otimes [L_k, U_k])$ and $([L_1, U_1] \oplus \ldots \oplus [L_k, U_k])$ to represent this generalization.

2.5 Combination functions

Suppose we wish to determine the probability that a single event e is true at time point t. Occasionally, we may have multiple sources of information where each source provides a different probability interval for e at time t. Combination functions are a generic mechanism to combine these intervals into a single interval.

Definition 2.10 (combination function). Let $S = \{[L_1, U_1], \ldots, [L_k, U_k]\}$ be a non-empty multiset of probabilistic intervals. Then a *combination function* χ is a function which takes S as input, and returns as output a probabilistic interval $[L, U]$ which satisfies the following axioms:
1. Identity: If $[L_1, U_1] = \ldots = [L_k, U_k]$, then $\chi(S) = [L_1, U_1]$. In other words, when all input intervals are equal, the output interval is also equal to all of the input intervals.
2. Bottomline: $L \leq \max\{L_i \mid [L_i, U_i] \in S\}$. In other words, the lower bound of the result cannot exceed the largest lower bound of the intervals in S. □

One may be tempted to add an axiom similar to Bottomline which applies to upper bounds. However, consider a case where $\cap_{[L_i, U_i] \in S}[L_i, U_i] = \emptyset$. In this case, it is reasonable for a human user to say "A conflict has occurred. In this case, I prefer to acknowledge being completely ignorant about the true, probabilities, i.e. I want to set $[L, U] = [0, 1]$." This seems like a reasonable strategy, but adding an extra axiom $U \geq \max\{U_i | [L_i, U_i] \in S\}$ would rule out this strategy. Similarly, $U \leq \min\{U_i | [L_i, U_i] \in S\}$ will be violated by a function which returns the closure of a union of intervals as the result. Finally, constraint $U \geq \min\{U_i | [L_i, U_i] \in S\}$, arguably the weakest and more reasonable, is violated by a function which returns interval $[0,0]$ whenever $\cap_{[L_i,U_i]\in S}[L_i,U_i]=\emptyset$.

Combination functions will be useful when describing TATA and TPA semantics for *intersection* (§5.2, §6.3) and *union* (§5.3, §6.4). For instance, after a union merges all tuples from two relations, the resulting relation may contain more than one tuple for a single event. Here, we could *compact* (merge) these tuples into a single tuple by applying a combination function.

Definition 2.11 (conflict). A multiset S of probability intervals *conflict* iff $\bigcap_{[L,U]\in S}[L,U]=\emptyset$. □

Note that all combination functions must find a way to remove conflicts. A class of combination functions called *equity combination functions* prescribe to the view that if $S=\{[L_1,U_1],[L_2,U_2]\}$ does not conflict, then $\chi(S)$ should equal $[L_1,U_1]\cap[L_2,U_2]$. However, if these intervals conflicted, then different equity combination functions may resolve the conflict in different ways.

Definition 2.12 (equity combination function). An equity combination function χ_e is a combination function where $(\bigcap_{[L,U]\in S}[L,U]\neq\emptyset)\Rightarrow(\chi_e(S)=\bigcap_{[L,U]\in S}[L,U])$. □

Example 2.5 (example equity combination functions). :

Name	Interval Returned when $\bigcap_{[L,U]\in S}[L,U]=\emptyset$
Optimistic Equity	$\chi_{eq}(S)=[\max(\{L_i \mid [L_i,U_i]\in S\}),\max(\{U_i \mid [L_i,U_i]\in S\})]$
Enclosing Equity	$\chi_{ec}(S)=[\min(\{L_i \mid [L_i,U_i]\in S\}),\max(\{U_i \mid [L_i,U_i]\in S\})]$
Pessimistic Equity	$\chi_{ep}(S)=[\min(\{L_i \mid [L_i,U_i]\in S\}),\min(\{U_i \mid [L_i,U_i]\in S\})]$
Rejecting Equity	$\chi_{er}(S)=[0,0]$
Skeptical Equity	$\chi_{esk}(S)=[0,1]$
Quasi-independence Equity	$\chi_{eqi}(S)=[\Pi_{[L_i,U_i]\in S}L_i,\Pi_{[L_i,U_i]\in S}U_i]$

Note that when $\bigcap_{[L,U]\in S}[L,U]\neq\emptyset$, all of the functions above return $\bigcap_{[L,U]\in S}[L,U]$.

Proposition 2. *Every function listed in Example 2.5 is an equity combination function.*

3. TP-RELATIONS

In this section, we define the syntax and semantics of a Temporal-Probabilistic relation. Intuitively, a TP-relation is a multiset of TP-tuples. Each TP-tuple consists of a "data" part and a "probabilistic-temporal" part. This latter part is called a TP-case statement and it intuitively specifies the probability with which the "data" part of the tuple is in the relation at different instances of time. Once TP-cases are defined in Section 3.1 below, we will provide a formal definition of TP-tuples and TP-relations in Sections 3.2 and 3.3.

We intend for TP-tuples defined in this section to represent *events*. An event is *instantaneous* if it can only occur at a single point in time. For example, consider the event "Toss *toss_id* of coin C comes up heads." This is an instantaneous event as it can only be true at a single point in time — the same coin cannot be tossed twice at the same time and two different tosses of the same coin represent two distinct events. Our TP-tuples will represent such *instantaneous events*. It is important to note that a real world event e (which has a continuous duration) may be modeled in our framework through two instantaneous events — the event $st(e)$ denoting the start of e and the event $end(e)$ denoting the end of e. Thus here, without loss of generality, we only consider events that are instantaneous. A similar assumption is made by Dyreson and Snodgrass[11].

3.1 TP-case statements

Definition 3.1 (TP-case statement over calendar τ). A *TP-case statement γ over calendar τ*, is an expression of the form

$$\boxed{\{\langle C_1, D_1, L_1, U_1, \delta_1\rangle, \ldots, \langle C_n, D_n, L_n, U_n, \delta_n\rangle\}}$$

where $n \geq 1$, C_i and D_i are a temporal constraints over τ, L_i and U_i are probabilities, δ_i is a distribution function over τ, and the following conditions are satisfied for all $1 \leq i \leq n$:
1. $(0 \leq L_i \leq U_i \leq 1)$.
2. $\text{sol}(C_i) \subseteq \text{sol}(D_i)$. This ensures that $\delta_i(D_i, t)$ is defined for each time point $t \in \text{sol}(C_i)$.
3. $|\text{sol}(C_i)| \geq 1$. In other words, C_i and D_i each have at least one solution in S_τ.
4. For all $1 \leq j \leq n$, $i \neq j \Rightarrow \text{sol}(C_i) \cap \text{sol}(C_j) = \emptyset$. In other words, $(C_i \wedge C_j)$ is always inconsistent. This ensures that each TP-case statement specifies at most one probability interval for each $t \in S_\tau$. Note that we do not have a similar requirement for $(D_i \wedge D_j)$.

For each $1 \leq i \leq n$, $\gamma_i = \langle C_i, D_i, L_i, U_i, \delta_i\rangle$ is called *a TP-case* of γ. On occasion, we may want to assign probabilities to every time point in S_τ. Here, $\text{sol}(C_n) = \text{sol}(\neg C_1 \wedge \neg C_2 \wedge \ldots \wedge \neg C_{n-1})$ and γ_n is called the *catch-all case*. For brevity, when γ_n is a catch-all case, we may use "$(*)$" to represent C_n.
Note: If $\text{sol}(C_i) = \text{sol}(D_i)$, we let "$(\#)$" be an abbreviation for C_i. □

One may wonder why *two* constraints (C_i and D_i) occur in each TP-case $\gamma_i = \langle C_i, D_i, L_i, U_i, \delta_i\rangle$. Intuitively, $\text{sol}(C_i)$ is the set of time points which γ_i is "interested in" while $\text{sol}(D_i)$ is the set of time points used when distributing the probability interval $[L_i, U_i]$ according to δ_i. When this TP-case is associated with a data-tuple d, it says that d is in some relation at some time point $t \in \text{sol}(C_i)$. The probability that d is true in the relation at such a t is $\delta_i(D_i, t)$. In other words, D_i is used to specify the set of time points used when distributing the probability interval $[L_i, U_i]$ according to δ_i. **This is an important distinction which is critically necessary.** Why ? Suppose that originally, $\text{sol}(C_i) = \text{sol}(D_i) = S = \{1, 2, 3, 4\}$ and $\delta_i =$ "$b, 0.5$". Thus, the probabilities associated with time points 1,2,3,4 are 0.125, 0.375, 0.375, 0.125. Now suppose we perform a *selection* operation (§5.4) which only asks for time points in the set $S' = \{2, 3\} \subset S$. If we had no D_i field in our TP-cases, then we would merely carry over the fact that S' has the binomial distribution on it. But applying this distribution to S' yields a probability of 0.5 to both 2 and 3 which is incorrect because selections should not change the probabilities assigned to time points $t \in S'$. Thus, some mechanism is needed to correctly compute the probabilities of relations resulting from algebraic operations executed.

Thus, in order to accurately compute probabilities, we must do one of two things:

(i) *either* carry with us the original set of values over which a probability distribution was defined, *or* (ii) determine how to accurately restrict an *arbitrary* distribution to apply to a subset of the set to which the distribution was originally applicable. The latter option requires a complex algebraic theory of distributions and its implementation is likely to be extremely expensive. For this reason, we chose the first option above.

For another (simpler) example, consider a TP-case statement with one TP-case γ_1:
$\{\langle(1/8/1997 \sim 5/8/1997), (1/8/1997 \sim 10/8/1997), 0.4, 0.8, u\rangle\}$ Intuitively, γ_1 says that some event occurred during the first five days of August 1997 (in other words, it occurred during one of the time points in $\mathrm{sol}(C_1)$). Since $\delta_1 =$ "u", the probability that it occurred on any of these days is the same. Specifically, this probability is $[\frac{1}{10} \cdot 0.4, \frac{1}{10} \cdot 0.8] = [0.04, 0.08]$ since we are uniformly distributing the probability mass $[0.4, 0.8]$ between all of the (10) time points in $\mathrm{sol}(D_1)$. In general, the probability interval for some time point $t \in \mathrm{sol}(C_i)$ is $[L_i \cdot \delta_i(D_i, t), U_i \cdot \delta_i(D_i, t)]$. Note that even when $(C_i = D_i)$, it is still possible that the probability interval for C_i, $[\sum_{t \in \mathrm{sol}(C_i)} \delta(D_i, t) \cdot L_i, \sum_{t \in \mathrm{sol}(C_i)} \delta(D_i, t) \cdot U_i]$, may not be equal to $[L_i, U_i]$ because δ can be *incomplete* (i.e., if $\sum_{t \in \mathrm{sol}(D_i)} \delta(D_i, t) < 1$).

Note: Even though TP-cases contain two distinct constraint fields, viz. C and D, this distinction can be hidden from the user, especially in base relations where C and D are equal.

The expression on the left below is a TP-case statement. However, the expression on the right is not a TP-case statement as the solution set to C_1 (and D_1) is empty.

$\{\langle(\#), (month < 6 \wedge year = 1997), 0.4, 0.8, g\rangle,$ $\langle(\#), (month \geq 6 \wedge year = 1997), 0.6, 0.6, u\rangle\}$	$\{\langle(\#), (year = 1996 \wedge year = 1997),$ $0.4, 0.8, g\rangle, \langle(\#), (year = 1998),$ $0.0, 0.0, u\rangle\}$

Furthermore, $\{\langle(\#), (month < 6 \wedge year = 1997), 0.4, 0.8, g\rangle, \langle(\#), (month \geq 3 \wedge year = 1997), 0.6, 0.6, u\rangle\}$
is not a TP-case statement as $(C_1 \wedge C_2)$ is not inconsistent (i.e., the probabilities for the overlapping time points are over-specified).

We reiterate that each temporal constraint in a TP-case statement must have a finite number of solutions (as S_τ is finite). We restrict ourselves to finite calendars and solution sets to avoid the complications which arise when trying to determine whether a constraint using negations is infinite or not.

3.2 TP-tuples

Before defining TP-tuples, the key concept of our framework, we need to define "hidden fields", which will serve to distinguish the identities of different TP-tuples.

Definition 3.2 (hidden field). A *hidden field* holds a lexicographically sorted *hidden list* of *field-value pairs* (i.e., "<field$_1$>:<value$_1$>, ..., <field$_n$>:<value$_n$>"). If there are no pairs to store, the hidden list will be EMPTY. □

Definition 3.3 (TP-tuple). Let $T_1 \sqsubseteq \cdots \sqsubseteq T_m$ be the linear hierarchy of time units over calendar τ and suppose $A = (A_1, \ldots, A_k)$ is a relational schema where for all $1 \leq i < k$, $A_i \notin \{$"C", "D", "L", "U", "δ", "L_t", "U_t", "H"$\}$, and for all $1 \leq j \leq m$, $A_i \neq T_j$. Furthermore, let A_k be the *hidden field* "H", let $d = (d_1, \ldots, d_k)$ be a (data) tuple over A, and let γ be a TP-case statement over τ.

Then $tp = (d, \gamma)$ is a *TP-tuple over relational schema A and calendar* τ. Intuitively, γ gives the probability for each $t \in S_\tau$ that d occurs at time t.

For instance, suppose our calendar consists of all days in 1996, and our relational schema is $A =$ (Item, Origin, Dest, H). Then

Item	Origin	Dest	H	C	D	L	U	δ
I1	Rome	Vienna		(#)	$day < 15 \wedge month = 11 \wedge$ $year = 1996$	0.5	0.6	u
				(#)	$day \geq 15 \wedge month = 11 \wedge$ $year = 1996$	0.4	0.4	u

is a TP-tuple which indicates that item "I1" left from "Rome" and will arrive in "Vienna" in November 1996 at some time before the 15th (with $50 - 60\%$ probability) or on/after the 15th (with 40% probability). This TP-tuple is not concerned with I1's arrival before or after November 1996 since no probabilities are assigned to this time range.

If we were sure that I1 did not arrive in Vienna before or after November 1996, we could add the TP-case $\langle (\#), (*), 0, 0, u \rangle$ to the TP-tuple above. If we had no information regarding I1's arrival before or after November 1996 but we were assuming that the distribution function for this time was "u", we could add the TP-case $\langle (\#), (*), 0, 1, u \rangle$ to the TP-tuple above.

Finally, if we had no information whatsoever regarding I1's arrival before or after November 1996, we would not change the TP-tuple above. Here, we are implicitly assigning a probability interval of $[0, 1]$ to each time point t (in 1996) which lies outside of November 1996 since for all $1 \leq i \leq n$, $t \notin \mathrm{sol}(C_i)$. We now introduce two intermediate operators that we use to define operators in the TP-algebra.

Definition 3.4 (manifest projection). Let $A = (A_1, \ldots, A_k)$ be a relational schema where A_k is the *hidden field* ("H") and let $d = (d_1, \ldots, d_k)$ be a (data) tuple over A. Then the *manifest projection of data tuple d*, denoted $\mathcal{P}(d)$, is defined as $(d_1, \ldots, d_{k-1})$. In other words, the tuple $\mathcal{P}(d)$ contains every value in d except the *hidden list* (d.H). Here, A_1 to A_{k-1} are known as *manifest data fields.* □

Definition 3.5 (hidden list concatenation). The *concatenation of hidden lists* d.H *and* d'.H, denoted (d.H $\|$ d'.H), is a hidden list h'' which can be constructed by lexicographically merging every field-value pair in d.H and d'.H. For instance if d.H = "Fld3:Val3, Fld6:Val6" and d'.H = "Fld4:Val4, Fld8:Val8, Fld9:Val9", then h'' = "Fld3:Val3, Fld4:Val4, Fld6:Val6, Fld8:Val8, Fld9:Val9". □

Intuitively, the manifest projection of a TP-relation simply eliminates the hidden field of the TP-relation while the concatenation of two hidden lists unites the contents of the two hidden lists, and then sorting them in lexicographic order. Note that in practice, multiple tables may use the same names for thier manifest data fields. To avoid confusion, an implementation should use "<TableName>.<FieldName>" instead of just a "<FieldName>" for each <field$_i$> in the hidden list.

3.3 TP-relations

Definition 3.6 (TP-relation). A *TP-relation over relational schema A and calendar τ*, denoted r, is a multiset [2] of TP-tuples over relational schema A and calendar τ.

A *base TP-relation* is a TP-relation which did not result from a query. In any base relation r the following should hold: for each TP-tuple $tp = (d, \gamma) \in r$, (i) $d.\mathrm{H} = \mathrm{EMPTY}$ and (ii) for each TP-case $\langle C_i, D_i, L_i, U_i, \delta_i \rangle \in \gamma$, $C_i = D_i$.

We associate with each TP-relation a *primary key*. This key will be used when we describe the TPA's semantics for projection (§6.8). □

Recall that a *primary key* is a minimal set of fields which, taken collectively, allow us to uniquely identify a tuple in a relation [20]. In the worst case, a primary key may need to contain every manifest data field in a relation. In practice, well designed databases use tuple ids, transaction ids, SSNs, timestamps, etc. to help keep the primary keys small.

Definition 3.7 (TP-database). A Probabilistic Temporal Database (abbreviated *TP-database*) over calendar τ is a pair $(Base, MView)$ where $Base$ is a set of base TP-relations over τ and $MView$ is a set of non-base TP-relations over τ. □

In base relations of a TP-Database, the contents of the hidden field will be EMPTY (since no fields have been projected out). For intermediate relations, the hidden field holds values of the form "<field>:<value>" for fields which have been projected out. Although these values should be hidden from the user, we shall see that they are important in determining whether two TP-tuples refer to the same event or not. For simplicity, we require all TP-relations in a TP-database to use the same calendar. Throughout this paper, we assume that all TP-relations are in the same TP-database.

3.4 Semantics and consistency of TP-relations

We are now ready to define the formal semantics of TP-relations. In order to provide such a semantics, we will extend classical logic [33] to the case of TP-relations, by extending the concept of an interpretation in classical logic [33] to handle TP-relations. Before doing this, a preliminary definition is needed.

Definition 3.8 (data-identical TP-tuples). TP-tuples $tp = (d, \gamma)$ and $tp' = (d', \gamma')$ are *data-identical* iff $(d = d')$. Note that tp and tp' may come from different TP-relations as long as both TP-relations have the same schema. Also, note that $(d = d')$ only if $(d.\mathrm{H} = d'.\mathrm{H})$. □

Recall that without loss of generality, we interpret TP-relations under the assumption that all data-identical TP-tuples refer to the same, unique event. If tp and tp' are data-identical, we assume that they provide complementary information for the same event. If tp and tp' are not data-identical, we assume that they refer to different events. Let $tp \in r$. Then $r[tp]$ denotes the multiset of all TP-tuples in r which are data-identical to tp. Since "data-identical" is a reflexive, symmetric,

[2] TP-relations are *multisets* instead of *sets* because they may contain two or more *distinct* copies of the same TP-tuple. We address this issue in more detail when we discuss compactness of TP-relations and compaction operations.

transitive relation on TP-tuples, it is also an equivalence relation on r where each $r[tp]$ corresponds to an equivalence class in this relation.

A TP-relation r is *compact* iff for each data tuple d and each time point t there is at most one TP-tuple $tp = (d, \gamma) \in r$ where $t \in \mathrm{sol}(C_1 \vee \ldots \vee C_n)$. Otherwise, as r contains at least two TP-tuples which refer to the same event at the same time, r is an *uncompact* TP-relation. Later, we will describe a variety of *compaction* operators which convert uncompact TP-relations into compact TP-relations by consolidating probabilistic information for each $r[tp] \subseteq r$ (cf., §6.3).

Intuitively, a TP-tuple $tp = (d, \gamma)$ is *consistent* if there exists a satisfying assignment of probabilities for each TP-case $\gamma_i \in \gamma$. This is given formal "teeth" through the following definition.

DEFINITION 3.9 (TP-INTERPRETATION). Let $A = (A_1, \ldots, A_k)$ be a relational schema, let τ be a calendar, and let $dom(A) = dom(A_1) \times \cdots \times dom(A_k)$ be the domain of A. Then a *TP-interpretation* over the pair A, τ is a function $I_{A,\tau}$: $dom(A) \times S_\tau \mapsto [0, 1]$ such that $(\forall d \in dom(A))(\sum_{t \in S_\tau} I_{A,\tau}(d, t) \leq 1)$. □

Let e be the event represented by data tuple d. Then $I_{A,\tau}(d, t) = p$ says that according to TP-interpretation $I_{A,\tau}$, the probability that e is true at time point t is p. Let D be a temporal constraint over τ. Then the *probability assigned by $I_{A,\tau}$ to D*, denoted $I_{A,\tau}(d, D)$, is equal to $\sum_{t \in \mathrm{sol}(D)} I_{A,\tau}(d, t)$. This intuition may be used to explain what it means for a TP-interpretation to satisfy a TP-tuple.

DEFINITION 3.10 (SATISFACTION). Let d be a tuple in relational schema A and let $\gamma_i = \langle C_i, D_i, L_i, U_i, \delta_i \rangle$ be a TP-case. Let $\varpi = \sum_{t \in sol(D_i)} \delta(D_i, t)$. Then $I_{A,\tau}$ *satisfies* $\langle d, \gamma_i \rangle$, denoted $I_{A,\tau} \models \langle d, \gamma_i \rangle$, iff the following conditions hold:
1.$L_i \cdot \varpi \leq I_{A,\tau}(d, D_i) \leq U_i \cdot \varpi$, i.e. the probability that $I_{A,\tau}$ assigns to D_i lies in the interval $\varpi[L_i, U_i]$.
2.$(\forall t \in \mathrm{sol}(C_i))(I_{A,\tau}(d, D_i) \cdot \delta_i(D_i, t) \cdot \varpi = I_{A,\tau}(d, t))$, i.e. $I_{A,\tau}$ distributes probabilities for each $t \in \mathrm{sol}(C_i)$ according to δ_i.
TP-interpretation $I_{A,\tau}$ *satisfies* TP-tuple $tp = (d, \gamma)$ ($I_{A,\tau} \models tp$), iff $I_{A,\tau} \models \langle d, \gamma_i \rangle$ for all $\gamma_i \in \gamma$. □

For example, let us reconsider the following TP-tuple.

Item	Origin	Dest	H	C	D	L	U	δ
I1	Rome	Vienna		(#)	$day < 15 \wedge month = 11 \wedge$ $year = 1996$	0.5	0.6	u
				(#)	$day \geq 15 \wedge month = 11 \wedge$ $year = 1996$	0.4	0.4	u

Consider the TP-interpretation defined as follows:
$I(\langle \text{I1,Rome,Vienna} \rangle, (d, 11, 1996)) = 0.04$ when $d < 15$.
$I(\langle \text{I1,Rome,Vienna} \rangle, (d, 11, 1996)) = \frac{0.4}{16}$ when $d \geq 15$.
$I(\langle \text{item,origin,dest} \rangle, (d, m, y)) = 0$ otherwise.

This TP-interpretation satisfies the TP-tuple above because
1. $I(\langle \text{I1,Rome,Vienna} \rangle, \{t \mid t.day < 15 \wedge t.month = 11 \wedge t.year = 1996\}) = 0.04 \cdot 14 = 0.56$
which lies between 0.5 and 0.6, and
2. $I(\langle \text{I1,Rome,Vienna} \rangle, \{t \mid t.day \geq 15 \wedge t.month = 11 \wedge t.year = 1996\}) = \frac{0.4}{16} \cdot 16 = 0.4$.

DEFINITION 3.11 (CONSISTENCY AND MUTUAL CONSISTENCY). A TP-tuple tp is *consistent* iff there exists a TP- interpretation $I_{A,\tau}$ such that $I_{A,\tau} \models tp$. A TP-relation r is *consistent* iff $(\exists I_{A,\tau})(\forall tp \in r)(I_{A,\tau} \models tp)$. TP-relations r and r' are *mutually consistent* iff $(\exists I_{A,\tau})((\forall tp \in r)(I_{A,\tau} \models tp) \wedge (\forall tp' \in r')(I_{A,\tau} \models tp'))$. Note that consistent TP-relations with different schemas must be mutually consistent. □

Later in this paper, we will provide algorithms to convert any TP-relation into a *compact* TP-relation. When a TP-relation is compact, there are no two TP-tuples that are data-identical, and hence, we can check consistency of a TP-relation by individually checking consistency of each TP-tuple. Suppose a TP-tuple $tp = (d, \gamma)$ has $\gamma = \{\langle C_1, D_1, L_1, U_1, \delta_1\rangle, \ldots, \langle C_n, D_n, L_n, U_n, \delta_n\rangle\}$ as its TP-case statement. Then it suffices to check that $\sum_{i=1}^{n}(L_i \cdot \Sigma_{t \in sol(C_i)} \delta_i(D_i, t)) \leq 1$.

If this condition holds, then the TP-tuple is consistent, as the formula above computes the sum of the lower bounds of probability for all timepoints t described in a TP-tuple. If this sum is less than or equal to 1, then an interpretation I which assigns to each timepoint t this lower bound will satisfy the TP-tuple, guaranteeing its consistency.

The rank of a tp-case $\gamma_i = \langle C_i, D_i, L_i, U_i, \delta_i\rangle$ is $rank(\gamma_i) = |C_i|$. Given a TP-tuple $tp = (d, \gamma); \gamma = \gamma_1, \ldots \gamma_k$, $rank(tp) = \sum_{i=1}^{k} rank(\gamma_i)$. Finally, the rank of a TP-relation r is given by: $rank(r) = \sum_{tp \in r} rank(tp)$. This allows us to state the following claim:

PROPOSITION 3. *Checking consistency of a compact TP-relation is linear in the rank of the TP-relation.*

4. ANNOTATED RELATIONS

An *annotated tuple* over a relational schema $\vec{A}$ is an expression of the form $(\vec{d}, t, L, U)$ where $\vec{d}$ is an (ordinary) tuple over relational schema $\vec{A}$, t is a time point, and L, U are real numbers in the $[0, 1]$ interval. An *annotated relation* is a finite set of annotated tuples. We will often use expressions such as $ann_r, ann_{r'}$, etc. to refer to annotated relations. An annotated tuple (at) provides probabilistic information ($[L, U]$) for one data tuple (d) at one point in time (t). Any tp-tuple can be converted into a set of annotated tuples as defined below.

DEFINITION 4.1 (ANNOTATED RELATION FOR A TP-TUPLE). Let $tp = (d, \gamma)$ be a TP-tuple over relational schema $(A_1, \ldots, A_k)$ and calendar τ where $d = (d_1, \ldots, d_k)$. Suppose γ contains n TP-cases of the form $\gamma_i = \langle C_i, D_i, L_i, U_i, \delta_i\rangle$ $(1 \leq i \leq n)$ and suppose τ consists of a linear hierarchy H containing m time units $T_1 \sqsubseteq \cdots \sqsubseteq T_m$. Here, each $t \in S_\tau$ will be of the form $t = (v_1, \ldots, v_m)$.

Then the *annotated relation for TP-tuple tp over calendar* τ, denoted ANN(tp), is defined as
$\{(d, t, L_t, U_t) \mid t \in \mathrm{sol}(C_i)$ for some $\gamma_i \in \gamma$ and $[L_t, U_t] = [L_i \cdot x, U_i \cdot x]$ where $x = \delta_i(D_i, t)\}$. □

Intuitively, in the above definition, x represents the percentage of sol(D_i)'s probability which is associated with time point t according to δ_i. Note that when we explicitly show all fields of an annotated tuple, $at = (d_1, \ldots, d_k, v_1, \ldots, v_m, L_t, U_t)$

is over the schema $(A_1,\ldots,A_k,T_1,\ldots,T_m,L_t,U_t)$. Here, A_1 to A_{k-1} are *manifest data fields*, A_k is the *hidden field*, T_1 to T_m are *temporal fields*, and L_t,U_t are *probabilistic fields*. Definition 4.1 can be easily extended to convert an annotated relation r into a TP-relation.

Definition 4.2 (annotated relation for a TP-relation). Let r be a TP-relation over τ containing n TP-tuples $tp_1\ldots tp_n$. Then the *annotated relation for TP-relation r over calendar τ*, denoted ANN(r), is defined as the multiset $(\mathrm{ANN}(tp_1)\uplus\ldots\uplus\mathrm{ANN}(tp_n))$ over τ, where $\uplus$ denotes *multiset union* operation.

It is clear see that ANN(tp) and ANN(r) can often be large and impractical to physically instantiate. This is why we **only use annotation for theoretical purposes such as illustrating a process or proving equivalences between query expressions. In our implementation, we never create annotated relations.**

The table below contains an annotated relation ANN(r) for a TP-relation r consisting of one TP-tuple $tp=(d,\gamma)$, where $d=$ ("D1", EMPTY) and $\gamma=\{\gamma_1\}$.
r:

Data	H	C	D	L	U	δ
D1		(#)	$day\leq 4\wedge month=11\wedge$ $year=1996$	0.4	0.8	u

$ANN(r)$:

Data	H	Day	Month	Year	L_t	U_t
D1		1	11	1996	0.1	0.2
D1		2	11	1996	0.1	0.2
D1		3	11	1996	0.1	0.2
D1		4	11	1996	0.1	0.2

However if γ_1's C_1 field was "$day\leq 3\wedge month=11\wedge year=1996$", then ANN($r$) would no longer contain the last tuple shown above. In general, note that changing C_i only affects the number of annotated tuples in ANN(r), not the probabilities for the remaining time points.

Notice the "0.1" and "0.2" values in the probabilistic fields above. These values were determined by uniformly distributing the available probability $[0.4,0.8]$ among the four annotated tuples in ANN(r). We were only justified in making this uniformity assumption since $\delta_1=$ "u". In general, TP-relations will only give us probability intervals for a range of time points, and determining (tight) probability intervals for each time point within that range requires us to apply a distribution function δ_i.

We associate with each annotated relation ann_r the *primary key* which is associated with r. This key will be used when we define the TATA's projection operation (§5.7). □

Note. Any annotated tuple (d,t,L,U) can be trivially converted into a tp-tuple (d,γ) where $\gamma=\gamma_1$, $\gamma_1=\langle C,D,L,U,\delta\rangle$, with $sol(C)=sol(D)=\{t\}$ and $\delta=$"u" (uniform). Therefore, any annotated relation ann_r has an associated trivial tp-relation r obtained by replacing each annotated tuple by its tp-conversion.

4.1 Semantics and consistency of annotated relations

Our semantics for annotated relations closely parallels our semantics for TP-relations (§3.4). Annotated tuples $at = (d, t, L_t, U_t)$ and $at' = (d', t', L'_t, U'_t)$ are called *data-identical* iff $(d = d')$. We interpret annotated relations under the assumption that all data-identical annotated tuples refer to the same event. If at and at' are not data-identical, we assume they refer to different events. Let d be a data tuple and let t be a time point. Then $\mathrm{ANN}(r)[d, t]$ denotes the equivalence class of the pair (d, t), i.e., the multiset of all $at \in \mathrm{ANN}(r)$ where $(at.d = d \wedge at.t = t)$.

Suppose $at = (d, t, L_t, U_t) \in \mathrm{ANN}(r)$, $at' = (d', t', L'_t, U'_t) \in \mathrm{ANN}(r)$, and $(d = d' \wedge t = t')$. Here, since $at, at' \in \mathrm{ANN}(r)$ refer to the same event at the same point in time, $\mathrm{ANN}(r)$ is an *uncompact* annotated relation. If there are no pairs of annotated tuples $at, at' \in \mathrm{ANN}(r)$ where $(d = d' \wedge t = t')$, then $\mathrm{ANN}(r)$ is a *compact* annotated relation. Later, we will describe a variety of *compaction* operators which convert uncompact annotated relations into compact annotated relations (e.g., §5.1). The following theorem states that the concept of "compact relation" for TP-relations and annotated relations coincide.

THEOREM 1. *A TP-relation r is compact iff its annotated counterpart,* $\mathrm{ANN}(r)$, *is compact.*

DEFINITION 4.3 (SATISFACTION OF ANNOTATED TUPLES). Let d be a tuple in relational schema A, let t be a time point in S_τ, and let $[L, U]$ be a probability interval. Then a TP-interpretation $I_{A,\tau}$ *satisfies* annotated tuple $at = (d, t, L_t, U_t)$, denoted $I_{A,\tau} \models at$, iff $L_t \le I_{A,\tau}(d, t) \le U_t$. □

DEFINITION 4.4 (CONSISTENCY OF ANNOTATED RELATIONS). An annotated tuple at is *consistent* iff $(\exists I_{A,\tau})(I_{A,\tau} \models at)$[3].An annotated relation $\mathrm{ANN}(r)$ is *consistent* iff $(\exists I_{A,\tau})(\forall at \in \mathrm{ANN}(r))(I_{A,\tau} \models at)$. Annotated relations $\mathrm{ANN}(r)$ and $\mathrm{ANN}(r')$ are *mutually consistent* iff $(\exists I_{A,\tau})((\forall at \in \mathrm{ANN}(r))(I_{A,\tau} \models at) \wedge (\forall at' \in \mathrm{ANN}(r'))(I_{A,\tau} \models at'))$. □

The following theorem tells us that any TP-interpretation satisfying TP-relation r also satisfies its annotation. A corollary of this is that if r is a consistent TP-relation, then $\mathrm{ANN}(r)$ is also consistent.

THEOREM 2. *If $I_{A,\tau} \models r$, then $I_{A,\tau} \models \mathrm{ANN}(r)$.*

COROLLARY 1. *If a TP-relation r is consistent, so is* $\mathrm{ANN}(r)$.

The converse of Theorem 2 is not true, i.e., it may be the case that a TP-interpretation satisfies $\mathrm{ANN}(r)$, but does not satisfy r. This is shown in the following example.

EXAMPLE 4.1 (SATISFACTION). Let r consist of one TP-tuple (d, γ) where $\gamma = \{\langle(\#), (1 \sim 2), 0.4, 0.8, u\rangle\}$. Then $\mathrm{ANN}(r) = \{at_1, at_2\}$ where $at_1 = (d, 1, 0.2, 0.4)$ and $at_2 = (d, 2, 0.2, 0.4)$.

Now consider the TP-interpretation $I_{A,\tau}$ such that $I_{A,\tau}(d, 1) = 0.3$ and $I_{A,\tau}(d, 2) = 0.4$. Clearly, $I_{A,\tau}$ satisfies $\mathrm{ANN}(r)$, but $I_{A,\tau} \not\models r$ because every TP-interpretation $J_{A,\tau}$ that satisfies r must have $J_{A,\tau}(d, 1) = J_{A,\tau}(d, 2)$.

[3]According to this definition, every annotated tuple $at = (d, t, L, U)$ is consistent as long as $L \le U$

This occurs since the details of the distribution get lost when annotating a relation — this is not surprising as annotated relations have no fields for including information about distributions. Instead, we can show that if r is compact, $(d, \gamma) \in r$, and $(d, t, L_t, U_t) \in \text{ANN}(r)$, then there must be a TP-interpretation $I_{A,\tau}$ of r such that $I_{A,\tau}(d,t) = L_t$ (i.e., $(\forall(d,t,L_t,U_t) \in \text{ANN}(r))(\exists I_{A,\tau})(I_{A,\tau} \models r \wedge I_{A,\tau}(d,t) = L_t)$). A similar statement applies to U_t. This means that the bounds contained in $\text{ANN}(r)$ are tight, and hence, $\text{ANN}(r)$ correctly captures the implied probability intervals for data tuple d at time t.

However, while the converse of Theorem 2 is not true, the converse to the Corollary is. Indeed,

Proposition 4. *Given a tp-relation r, if $ANN(r)$ is consistent, then so is r.*

Returning back to the problem of reachability of the bounds we note that for lower bounds of compact TP-relations one can make a stronger claim than the one stated above.

Theorem 3. *Let r be a compact TP-relation containing a TP-tuple (d, γ), and suppose $(d, t, L_t, U_t) \in \text{ANN}(r)$. Then there is a TP-interpretation $I_{A,\tau}$ satisfying r such that $I_{A,\tau}(d,t) = L_t$.*

Theorems 2 and 3 jointly tell us that as far as lower bounds are concerned, r and $\text{ANN}(r)$ are equivalent when r is known to be compact. Later, we will describe mechanisms to make a TP-relation r compact.

Note that, unlike the $\forall\exists$ statement in the example above, the $\exists\forall$ statement of the threorem does not hold for upper bounds — the reason for this is that in a TP-tuple, the upper bounds may often be loose (i.e., not tight). For instance, consider the following TP-tuple:

Data	H	C	D	L	U	δ
D1		(#)	(5/1/1998)	0.6	1	u
		(#)	(6/1/1998)	0.4	1	u

It is easy to see that the upper bounds of the TP-cases above can be tightened to 0.6 and 0.4 respectively. Hence, these upper bounds are “loose” and need to be tightened if a theorem similar to Theorem 3 is to hold.

Definition 4.5 (tightening). Let $tp = (d, \gamma)$, $\gamma = \gamma_1, \ldots, \gamma_n$, where $\gamma_i = \langle C_i, D_i, U_i, L_i, \delta_i \rangle$ be a tp-tuple. A *tightening* of tp is a tp-tuple $tp' = (d, \gamma')$, $\gamma' = \gamma'_1, \ldots, \gamma'_m$, $\gamma'_i = \langle C'_i, D'_i, U'_i, L'_i, \delta'_i \rangle$, such that
1. $\cup_{i=1}^{n} C_i = \cup_{i=i}^{m} C'_i$.
2. For all TP-interpretations $I_{A,\tau}$, $I_{A,\tau} \models (d, \gamma)$ iff $I_{A,\tau} \models (d, \gamma')$.
3. For all $t \in \cup_{i=1}^{n} C_i$, let $(d, t, L, U) \in ANN(tp)$ and $(d, t, L', U') \in ANN(tp')$. Then $L' = L$ and $U' \leq U$.

A TP-tuple (d, γ) is said to be *tight* iff there is no other TP-tuple (d, γ') such that: (i) (d, γ') is a tightening of (d, γ) and (ii) for all TP-interpretations $I_{A,\tau}$, $I_{A,\tau} \models (d, \gamma)$ iff $I_{A,\tau} \models (d, \gamma')$.

A TP-relation is *tight* iff every TP-tuple in it is tight. □

Theorem 4. *Let r be a compact, tight, TP-relation containing a TP-tuple (d, γ), and suppose $(d, t, L_t, U_t) \in \text{ANN}(r)$. Then there is a TP-interpretation $I_{A,\tau}$ satisfying r such that $I_{A,\tau}(d,t) = U_t$.*

Theorems 3 and 4 jointly tell us that the conversion of a TP-relation r to annotated form preserves bounds when r is tight and compact. Every TP-relation can be tightened using the following algorithm:

Algorithm Tighten-TP-Tuple(tp):
Input: TP-tuple $tp = (d, \gamma)$ where $\gamma = \{\gamma_1, \ldots, \gamma_n\}$ and for all $1 \leq i \leq n$, $\gamma_i = \langle C_i, D_i, L_i, U_i, \delta_i \rangle$
Output: Tight TP-tuple tp'' which is a tightening of tp
Note: In this algorithm, let $\delta(D, C)$ be a "shortcut" for the following expression: $\Sigma_{t \in \mathrm{sol}(C)} \delta(D, t)$

$L := 0$; $U := 0$; `// ` $[L, U]$ ` will hold the sum of the lower and upper bounds`
for $i := 1$ **to** n **do** {
  $L'_i := \delta_i(D_i, C_i) \cdot L_i$; $L := L + L'_i$;
  $U'_i := \delta_i(D_i, C_i) \cdot U_i$; $U := U + U'_i$; }
if $U \leq 1.0$ **then return** $tp'' := tp$; `// If` $U \leq 1.0$`, then` tp `was already tight`
$\gamma'' = \emptyset$;
for $i := 1$ **to** n **do** {
  if $\delta_i \neq u$ **then** {
    foreach $t \in \mathrm{sol}(C_i)$ {
      $L_t := \delta_i(D_i, t) \cdot L_i$; $U_t := \delta_i(D_i, t) \cdot U_i$; $U' := 1 - (L - L_t)$;
      if $U' < U_t$ **then** $U_t := U'$;
      Add TP-case $\langle (\#), (t), L_t, U_t, u \rangle$ to γ''; } }
  else {
    $m = |\mathrm{sol}(D_i)|$; $L_t = \frac{L_i}{m}$; $U_t = \frac{U_i}{m}$;
    $U' := 1 - (L - L_t)$; $U''_i := m \cdot \min(U_t, U')$;
    Add TP-case $\langle C_i, D_i, L_i, U''_i, \delta_i \rangle$ to γ''; } }
return $tp'' := (d, \gamma'')$;
End-Algorithm

4.2 Sample annotated relations

Let r consist of one TP-tuple which contains two TP-cases as shown below.

Item	Origin	Dest	H	C	D	L	U	δ
I1	Rome	Paris		(#)	$day \leq 2 \wedge month = 8$ $year = 1997$	0.5	0.7	Φ
				(#)	$day \geq 5 \wedge day \leq 7 \wedge$ $month = 8 \wedge year = 1997$	0.3	0.6	Φ

Note that the variable Φ must be instantiated. If $\Phi =$ "u", ANN(r) will be

Item	Origin	Dest	H	Day	Month	Year	L_t	U_t
I1	Rome	Paris		1	8	1997	0.25	0.35
I1	Rome	Paris		2	8	1997	0.25	0.35
I1	Rome	Paris		5	8	1997	0.10	0.20
I1	Rome	Paris		6	8	1997	0.10	0.20
I1	Rome	Paris		7	8	1997	0.10	0.20

where $[L_t, U_t] = \frac{1}{2} \cdot [0.5, 0.7]$ for the first two tuples and $[L_t, U_t] = \frac{1}{3} \cdot [0.3, 0.6]$ for the remaining tuples in ANN(r). However if $\Phi =$ "g", ANN(r) will be

Item	Origin	Dest	H	Day	Month	Year	L_t	U_t
I1	Rome	Paris		1	8	1997	0.25	0.35
I1	Rome	Paris		2	8	1997	0.125	0.175
I1	Rome	Paris		5	8	1997	0.15	0.30
I1	Rome	Paris		6	8	1997	0.075	0.15
I1	Rome	Paris		7	8	1997	0.0375	0.075

where $[L_t, U_t] = \frac{1}{2} \cdot [0.5, 0.7]$, $\frac{1}{4} \cdot [0.5, 0.7]$, $\frac{1}{2} \cdot [0.3, 0.6]$, $\frac{1}{4} \cdot [0.3, 0.6]$, and $\frac{1}{8} \cdot [0.3, 0.6]$ for the first through fifth tuples of ANN(r) respectively. Notice that modifying Φ (i.e., the distribution function δ) only affects the L_t and U_t fields of ANN(r).

5. THEORETICAL ANNOTATED TEMPORAL ALGEBRA

In this section, we define the Theoretical Annotated Temporal Algebra and provide definitions for *compac- tion*, *intersection*, *union*, *selection*, *difference*, *cartesian product*, *projection*, and *join* on annotated relations.

We know that every TP-relation can be converted into a (potentially very large) annotated relation. As annotated relations are *explicit* representations of TP-relations, the definition of the above operations on annotated relations can be explicitly defined and justified — this is what we do in this section. Then, in Section 6, we will show how these operations can be implemented in the TP-Algebra in such a way that the TP-Algebra operations efficiently implement the annotated algebra operations on the *implicit* (smaller) TP-relations, rather than their larger annotated counterparts, this avoiding the need for explicit annotated relations altogether.

The definitions in this section will produce a new annotated relation ann''_r based on input from consistent annotated relations ann_r, and ann'_r. Oftentimes, these definitions will refer to annotated tuples at, at' which are assumed to be of the form $at = (d, t, L_t, U_t)$ and $at' = (d', t', L'_t, U'_t)$.

Note: Our examples illustrating the Theoretical Annotated Temporal Algebra and the TP-Algebra will be based on the relations shown in Figure 3.

5.1 Compaction of an annotated relation

The first operation we define will be *compaction* as this operator is needed to define other operators. Compaction is the TP analog of duplicate elimination in the relational algebra.

DEFINITION 5.1 (COMPACTION OF AN ANNOTATED RELATION). A function κ from annotated relations to annotated relations is called a *compaction operation* if it satisfies the following axioms:

- Compactness : $\kappa(ann_r)$ is *compact* for all annotated relations ann_r.
- No Fooling Around (NFA) : If ann_r is compact, then $\kappa(ann_r) = ann_r$.
- Conservativeness : If $at = (d, t, L_t, U_t) \in \kappa(ann_r)$, then $\exists at' = (d, t, L'_t, U'_t) \in ann_r$. □

The Compactness axiom assures us that the result of a compaction operation will be a compact relation. The NFA axiom states that applying compaction operation to a compact relation should not change the relation. The Compactness and NFA axioms jointly guarantee that compaction operations are idempotent, i.e.

r_1

Data	H	C	D	L	U	δ
D1		(#)	(1/8/1997 ∼ 3/8/1997)	0.64	0.88	g
		(#)	(5/8/1997 ∼ 8/8/1997)	0.40	0.80	u

ANN(r_1)

Data	H	Day	Month	Year	L_t	U_t
D1		1	8	1997	0.32	0.44
D1		2	8	1997	0.16	0.22
D1		3	8	1997	0.08	0.11
D1		5	8	1997	0.10	0.20
D1		6	8	1997	0.10	0.20
D1		7	8	1997	0.10	0.20
D1		8	8	1997	0.10	0.20

r_2

Data	H	C	D	L	U	δ
D1		(#)	(2/8/1997 ∼ 3/8/1997)	0.20	0.50	g
		(#)	(6/8/1997 ∼ 9/8/1997)	0.40	0.80	u

ANN(r_2)

Data	H	Day	Month	Year	L_t	U_t
D1		2	8	1997	0.10	0.25
D1		3	8	1997	0.05	0.125
D1		6	8	1997	0.10	0.20
D1		7	8	1997	0.10	0.20
D1		8	8	1997	0.10	0.20
D1		9	8	1997	0.10	0.20

r_3

Data1	Data2	H	C	D	L	U	δ
D1	D2		(#)	(2/8/1997 ∼ 2/8/1997)	0.20	0.40	u
D1	D3		(#)	(2/8/1997 ∼ 3/8/1997)	0.60	0.80	g

ANN(r_3)

Data1	Data2	H	Day	Month	Year	L_t	U_t
D1	D2		2	8	1997	0.20	0.40
D1	D3		2	8	1997	0.30	0.40
D1	D3		3	8	1997	0.15	0.20

r_4

Data1	Data2	H	C	D	L	U	δ
D1	D2		(#)	(2/8/1997 ∼ 3/8/1997)	0.20	1.00	g
D1	D3		(#)	(3/8/1997 ∼ 3/8/1997)	0.50	0.50	u
D4	D5		(#)	(1/8/1997 ∼ 1/8/1997)	0.70	0.80	u

ANN(r_4)

Data1	Data2	H	Day	Month	Year	L_t	U_t
D1	D2		2	8	1997	0.10	0.50
D1	D2		3	8	1997	0.05	0.25
D1	D3		3	8	1997	0.50	0.50
D4	D5		1	8	1997	0.70	0.80

Figure 3: Example Base TP and Annotated Relations

$\kappa(\kappa(ann_r)) = \kappa(ann_r)$. The Conservativeness axiom says that any information which appears in the result of a compaction has to originate from information in the initial relation; no information about "new" events, or events at "new" time points gets added during compaction.

It should be clear that there are many possible ways to compact a relation. One possible class of compaction strategies involves the use of a combination function (as defined in Section 2.5).

Definition 5.2 (χ-compaction of an annotated relation). Let χ be a combination function. Then the *χ-compaction of annotated relation ann_r*, denoted $\kappa_\chi(ann_r)$, is defined as $\kappa_\chi(ann_r) =$

$\{at = (d,t,L_t,U_t) \mid [L_t,U_t] = \chi(\{[L_1^{(d,t)},U_1^{(d,t)}],\dots,[L_k^{(d,t)},U_k^{(d,t)}]\})\}$ where $ann_r[d,t] = \{at_1^{(d,t)},\dots,at_k^{(d,t)}\}$ and $at_i^{(d,t)} = (d,t,L_i^{(d,t)},U_i^{(d,t)})$. □

Intuitively, in combination function based compactions, χ is applied to the multiset of all $[L_i,U_i]$s associated with (d,t). The resulting $[L,U]$ then becomes the only probability interval associated with (d,t). The following proposition states that an operation defined in this manner is indeed a *compaction operation*.

To simplify notation, whenever we have a combination function χ_s, instead of denoting the corresponding compaction operation κ_{χ_s} we will write κ_s. This way, e.g. instead of $\kappa_{\chi_{eq}}$ we write κ_{eq}. This rule will be also used for other operations.

Proposition 5. *Let χ be any combination function. Then κ_χ is a compaction operation.*

Theorem 5. *If $at' = (d,t,L'_t,U'_t) \in ann_r$, then $\exists at = (d,t,L_t,U_t) \in \kappa_\chi(ann_r)$.*

Theorem 5 indicates that every tuple in ann_r leads to a corresponding tuple in compaction for combination function based compaction operations.

Another possible class of compaction strategies uses *p-strategies* (i.e., probabilistic conjunction or disjunction strategies as defined in Section 2.4). These compactions, denoted κ_ρ, are defined in the same way as $\kappa_\chi(ann_r)$ except we let $[L_t,U_t] = ([L_1^{(d,t)},U_1^{(d,t)}] \otimes_\rho \dots \otimes_\rho [L_k^{(d,t)},U_k^{(d,t)}])$ when ρ is a conjunctive p-strategy, and let $[L_t,U_t] = ([L_1^{(d,t)},U_1^{(d,t)}] \oplus_\rho \dots \oplus_\rho [L_k^{(d,t)},U_k^{(d,t)}])$ when ρ is a disjunctive p-strategy.

Proposition 6. *Let ρ be any p-strategy. Then $\kappa_\rho(ann_r)$ is a compaction operation.*

5.2 Intersection of two annotated relations

The intersection of annotated relations ann_r and ann'_r extracts information common to both relations. In our algebra, we break intersection into two suboperations: First, a *multiset intersection* will extract all tuples from both ann_r and ann'_r which contain "common information". Then, we will use one of our previously-defined compaction operators to compact the result of this multiset intersection. Finally, *intersection* will be defined as a combination of these suboperations. Note that intersection (and multiset intersection) is only defined when both relations have the same schema.

Definition 5.3 (multiset intersection of two annotated relations). The *multiset intersection of annotated relations ann_r and ann'_r*, denoted $ann_r \cap$

ann'_R, is defined as $ann''_r = \{at \in ann_r \mid (\exists at' \in ann''_r)(d = d' \wedge t = t')\} \cup \{at' \in ann'_r \mid (\exists at \in ann_r)(d = d' \wedge t = t')\}$. □

Intuitively, ann''_r contains all $at \in ann_r$ and all $at' \in ann'_r$ where at and at' refer to the same event at the same point in time. Recall that "$(\exists at \in ann_r)$" and "$(\exists at' \in ann'_r)$" are shorthand for "$(\exists(d, t, L_t, U_t) \in ann_r)$" and "$(\exists(d', t', L'_t, U'_t) \in ann'_r$." respectively. For greater clarity and conciseness, our definitions will make use of this implicit notation. First table below shows $ann''_r = \mathrm{ANN}(r_1) \cap \mathrm{ANN}(r_2)$

$\mathrm{ANN}(r_1) \cap \mathrm{ANN}(r_2)$						
Data	H	Day	Month	Year	L_t	U_t
D1		2	8	1997	0.16	0.22
D1		3	8	1997	0.08	0.11
D1		6	8	1997	0.10	0.20
D1		7	8	1997	0.10	0.20
D1		8	8	1997	0.10	0.20
D1		2	8	1997	0.10	0.25
D1		3	8	1997	0.05	0.125
D1		6	8	1997	0.10	0.20
D1		7	8	1997	0.10	0.20
D1		8	8	1997	0.10	0.20

$\mathrm{ANN}(r_1) \cap_{eq} \mathrm{ANN}(r_2) = \kappa_{eq}(ann''_r)$						
Data	H	Day	Month	Year	L_t	U_t
D1		2	8	1997	0.16	0.22
D1		3	8	1997	0.08	0.11
D1		6	8	1997	0.10	0.20
D1		7	8	1997	0.10	0.20
D1		8	8	1997	0.10	0.20

Clearly, ann''_r above is uncompacted. To obtain a compact annotated relation, we may use any κ_χ compaction operator. Using this operator when defining intersection makes sense because r, r' may both contain data tuple d at some time point t, but with different probabilities. In this case we are combining two different probabilities assigned to the same event by two different sources (relations ann_r and ann'_r). This is exactly what combination functions χ were designed to support. The table on the right above shows the result of this computation.

Definition 5.4 (intersection of two annotated relations). The *intersection of annotated relations ann_r and ann'_r under the χ combination function*, denoted $ann_r \cap_\chi ann'_r$, is defined as $\kappa_\chi(ann_r \cap ann'_r)$. □

5.3 Union of two annotated relations

Just like intersection, the union of two annotated relations will be presented as a combination of two suboperations: multiset union, which combines the information from two relations together and compaction, which compacts the result. As always, union is only defined when both relations have the same schema.

Definition 5.5 (multiset union of two annotated relations). The *mul-*

tiset union of annotated relations
ann_r *and* ann'_r, denoted $ann_r \cup ann'_r$, is defined as $ann''_r = ann_r \uplus ann'_r$. □

Intuitively, ann''_r will contain all $at \in ann_r$ and all $at' \in ann'_r$. As in the case of intersection, ann''_r may over-specify probabilistic information. We can consolidate this information by using a κ_χ compaction operator. The reason for using the operator κ_χ instead of a conjunction strategy is exactly for the same reason that we used the κ_χ compaction operator when defining intersection (see discussion preceding Definition 5.4).

Definition 5.6 (union of two annotated relations). The *union of annotated relations* ann_r *and* ann'_r *under the* χ *combination function*, denoted $ann_r \cup_\chi ann'_r$, is defined as $\kappa_\chi(ann_r \cup ann'_r)$. □

The tables below show the results of $ann''_r = \mathrm{ANN}(r_1) \cup \mathrm{ANN}(r_2)$ and $\mathrm{ANN}(r_1) \cup_{eq} \mathrm{ANN}(r_2)$.

$\mathrm{ANN}(r_1) \cup \mathrm{ANN}(r_2)$						
Data	H	Day	Month	Year	L_t	U_t
D1		1	8	1997	0.32	0.44
D1		2	8	1997	0.16	0.22
D1		3	8	1997	0.08	0.11
D1		5	8	1997	0.10	0.20
D1		6	8	1997	0.10	0.20
D1		7	8	1997	0.10	0.20
D1		8	8	1997	0.10	0.20
D1		2	8	1997	0.10	0.25
D1		3	8	1997	0.05	0.125
D1		6	8	1997	0.10	0.20
D1		7	8	1997	0.10	0.20
D1		8	8	1997	0.10	0.20
D1		9	8	1997	0.10	0.20

$\mathrm{ANN}(r_1) \cup_{eq} \mathrm{ANN}(r_2) = \kappa_{eq}(ann''_r)$						
Data	H	Day	Month	Year	L_t	U_t
D1		1	8	1997	0.32	0.44
D1		2	8	1997	0.16	0.22
D1		3	8	1997	0.08	0.11
D1		5	8	1997	0.10	0.20
D1		6	8	1997	0.10	0.20
D1		7	8	1997	0.10	0.20
D1		8	8	1997	0.10	0.20
D1		9	8	1997	0.10	0.20

Note that although $\mathrm{ANN}(r_1)$ and $\mathrm{ANN}(r_2)$ are both *consistent*, $\mathrm{ANN}(r_1) \cup_{eq} \mathrm{ANN}(r_2)$ above is an *inconsistent* annotated relation (since for some data tuple d, the sum of the L_t values exceeds 1.0). This occurs because $\mathrm{ANN}(r_1)$ and $\mathrm{ANN}(r_2)$ are not *mutually consistent* (Def. 4.4). In general, if consistent annotated relations

ANN(r) and ANN(r') are also mutually consistent, then ANN(r) $\cup_{eq}$ ANN(r') will always be consistent.

5.4 Selection on an annotated relation

We represent a *selection condition over calendar* τ by the symbol $\mathcal{C}$. If $\mathcal{C}$ is of the form $(F\ op\ v)$ or $(t_1 \sim t_2)$, then $\mathcal{C}$ is an *atomic condition over* τ. Let $\mathcal{C}$ be an atomic condition, let $T_1 \sqsubseteq \cdots \sqsubseteq T_m$ be a linear hierarchy H of time units over τ, and suppose TP-relation r is over relational schema $A = (A_1, \ldots, A_k)$. Then one of the following cases must hold:
- If $F = A_i$ for some $1 \leq i < k$, then $\mathcal{C}$ is a *data condition.*
- If $F = T_j$ for some time unit T_j in H or if $\mathcal{C}$ is of the form $(t_1 \sim t_2)$, then $\mathcal{C}$ is a *temporal condition.*
- If $F =$ "L" or $F =$ "U", then $\mathcal{C}$ is a *probabilistic condition.*
- Otherwise, $\mathcal{C}$ is an *inapplicable condition.* In this case, $\sigma_{\mathcal{C}}(r)$ and $\sigma_{\mathcal{C}}(\text{ANN}(r))$ are not defined. Notice that selections on the *hidden field* (i.e., $F = A_k$) are not permitted. Throughout this paper, we will assume that $\mathcal{C}$ is not an inapplicable condition.

Definition 5.7 (selection on an annotated relation; atomic condition). The *selection of atomic condition* $\mathcal{C}$ *on annotated relation* ann_r, denoted $\sigma_{\mathcal{C}}(ann_r)$, is defined in the following way:
- If $\mathcal{C}$ is a data condition, $\sigma_{\mathcal{C}}(ann_r) = \{at \in ann_r \mid d \text{ satisfies } \mathcal{C}\}$. In this case, our selection is based on the classical relational algebra.
- If $\mathcal{C}$ is a temporal condition, $\sigma_{\mathcal{C}}(ann_r) = \{at \in snn_r \mid t \in \text{sol}(\mathcal{C})\}$.
- If $\mathcal{C}$ is a probabilistic condition, $\sigma_{\mathcal{C}}(ann_r) = \{at \in ann_r \mid ([L, U] = [L_t, U_t]) \text{ satisfies } \mathcal{C}\}$. □

For example if $\mathcal{C} = (2/8/1997 \sim 7/8/1997)$, $\sigma_{\mathcal{C}}(\text{ANN}(r_1))$ and $\sigma_{\mathcal{C}}(\text{ANN}(r_2))$ will be

$\sigma_{\mathcal{C}}(\text{ANN}(r_1))$

Data	H	Day	Month	Year	L_t	U_t
D1		2	8	1997	0.16	0.22
D1		3	8	1997	0.08	0.11
D1		5	8	1997	0.10	0.20
D1		6	8	1997	0.10	0.20
D1		7	8	1997	0.10	0.20

$\sigma_{\mathcal{C}}(\text{ANN}(r_2))$

Data	H	Day	Month	Year	L_t	U_t
D1		2	8	1997	0.10	0.25
D1		3	8	1997	0.05	0.125
D1		6	8	1997	0.10	0.20
D1		7	8	1997	0.10	0.20

but if $\mathcal{C} = (L \neq 0.10)$, $\sigma_{\mathcal{C}}(\text{ANN}(r_1))$ and $\sigma_{\mathcal{C}}(\text{ANN}(r_2))$ will be

$\sigma_{\mathcal{C}}(\mathrm{ANN}(r_1))$						
Data	H	Day	Month	Year	L_t	U_t
D1		1	8	1997	0.32	0.44
D1		2	8	1997	0.16	0.22
D1		3	8	1997	0.08	0.11

$\sigma_{\mathcal{C}}(\mathrm{ANN}(r_2))$						
Data	H	Day	Month	Year	L_t	U_t
D1		3	8	1997	0.05	0.125

Later, we will describe how to perform selections with non-atomic selection conditions (§6.5).

5.5 Difference of two annotated relations

As in the classical relational algebra, difference is only defined when both relations have the same schema. There are many possible ways of defining difference, but we have chosen to base our definition on the intuition that if two relations r and r' represent the information that two different "agents" have about the same world, then $r - r'$ should represent the information about the world that r has and r' does not.

Definition 5.8 (difference of two annotated relations). The *difference* $ann_r - ann'_r$ *of annotated relations* ann_r *and* ann'_r is $ann_r - ann'_r = \{at \in ann_r \mid (\forall at' \in ann'_r)(d \neq d' \vee t \neq t')\}$. □

Thus, ann''_r will not include $at \in ann_r$ if there exists an $at' \in ann'_r$ which refers to the same event at the same point in time. For example, $\mathrm{ANN}(r_1) - \mathrm{ANN}(r_2)$ and $\mathrm{ANN}(r_2) - \mathrm{ANN}(r_1)$ will be

$\mathrm{ANN}(r_1) - \mathrm{ANN}(r_2)$						
Data	H	Day	Month	Year	L_t	U_t
D1		1	8	1997	0.32	0.44
D1		5	8	1997	0.10	0.20

$\mathrm{ANN}(r_2) - \mathrm{ANN}(r_1)$						
Data	H	Day	Month	Year	L_t	U_t
D1		9	8	1997	0.10	0.20

Suppose $at_1 = (d, t, 0.2, 0.4) \in ann_r$ and $at_2 = (d, t, 0, 1) \in ann'_r$. Then by definition, ann''_r will not contain at_1. Now suppose we removed from ann'_r all annotated tuples where $[L'_t, U'_t] = [0, 1]$. Here, ann''_r will contain at_1. Apparently, we cannot simply throw out tuples where $[L'_t, U'_t] = [0, 1]$.

Intuitively, if we do not have an annotated tuple for data tuple d at time t, then "we do not know anything about (d, t)'s probability". In this case, (d, t) is implicitly assigned a probability interval of $[0, 1]$. On the other hand, at_2 indicates that "we know that we do not know anything about (d, t)'s probability". This distinction is subtle yet important; by keeping these two cases distinct, we allow both the *closed world assumption* (where (d, t) is implicitly assigned a probability interval of $[0, 0]$) and the *open world assumption*.

5.6 Cartesian product of two annotated relations

Each tuple in the result of a cartesian product reflects the conjunction of two events. Suppose that at time t, events e_1 and e_2 have probability intervals $[L_1, U_1]$ and $[L_2, U_2]$ respectively. In order to compute the probability interval $[L, U]$ for the event $(e_1 \wedge e_2)$ at time t, we must apply a *probabilistic conjunction strategy* α, i.e., $[L, U] = [L_1, U_1] \otimes_\alpha [L_2, U_2]$ (§2.4). This allows users to ask queries such as "Compute the cartesian product of annotated relations ann_r and ann'_r under the assumption that there is no information about dependencies between events in these relations."

Definition 5.9 (cartesian product of two annotated relations). The *cartesian product of annotated rela- tions ann_r and ann'_r under the α probabilistic conjunction strategy*, denoted $ann_r \times_\alpha ann'_r$, is defined as $ann''_r = \{(d'', t, L''_t, U''_t) \mid (\exists at \in ann_r) \wedge (\exists at' \in ann'_r) \wedge$
$(d'' = (\mathcal{P}(d), \mathcal{P}(d'), h'')) \wedge (h'' = (d.\mathrm{H} \parallel d'.\mathrm{H})) \wedge (t = t') \wedge ([L''_t, U''_t] = [L_t, U_t] \otimes_\alpha [L'_t, U'_t])\}$. □

Note that cartesian products only combine annotated tuples which refer to the same time point. It computes the combined data tuple d'' by merging (i) manifest data fields from ann_r (i.e., $\mathcal{P}(d)$), (ii) manifest data fields from ann'_r (i.e., $\mathcal{P}(d')$), and (iii) $h'' = d.\mathrm{H} \parallel d'.\mathrm{H}$ (i.e., the hidden list concatenation of d.H and d'.H). It then computes the combined probability interval by applying user selected conjunction strategy α. This is highly appropriate because when computing Cartesian Products, we are looking at the probability that the concatenation of the two data tuples is in the (ordinary set theoretic) cartesian product of the two relations at a given instant of time. This is therefore a *conjunctive* event, and hence, the use of a conjunctive p-strategy when performing cartesian products.

For example,

ANN(r_1) $\times_{ig}$ ANN(r_2)							
r_1.D	r_2.D	H	Day	Month	Year	L_t	U_t
D1	D1		2	8	1997	0.00	0.22
D1	D1		3	8	1997	0.00	0.11
D1	D1		6	8	1997	0.00	0.20
D1	D1		7	8	1997	0.00	0.20
D1	D1		8	8	1997	0.00	0.20

ANN(r_1) $\times_{pc}$ ANN(r_2)							
r_1.D	r_2.D	H	Day	Month	Year	L_t	U_t
D1	D1		2	8	1997	0.10	0.22
D1	D1		3	8	1997	0.05	0.11
D1	D1		6	8	1997	0.10	0.20
D1	D1		7	8	1997	0.10	0.20
D1	D1		8	8	1997	0.10	0.20

5.7 Projection on an annotated relation

A list $\mathcal{F}$ of fields is said to be *projectable* w.r.t. TP-relation r if (i) every field in $\mathcal{F}$ is a manifest data field of r, and (ii) $\mathcal{F}$ is non-empty. $\mathcal{F}$ is *projectable* w.r.t. annotated relation ANN(r) iff $\mathcal{F}$ is *projectable* w.r.t. r. It is important to note that hidden fields cannot be projected out.

DEFINITION 5.10 (PROJECTION ON AN ANNOTATED RELATION). Let $\mathcal{F}$ be a list of fields which are *projectable* w.r.t. ann_r and let "$A_1, \ldots, A_n$" be the (possibly empty) list of all manifest data fields which appear in the primary key of ann_r but do not appear in $\mathcal{F}$. Then the *projection of field list $\mathcal{F}$ on annotated relation ann_r*, denoted $\pi_{\mathcal{F}}(\mathrm{ANN}(r))$, is defined as $ann''_r = \{(d'', t, L_t, U_t) \mid (\exists at \in a) \wedge (d'' = (\pi_{\mathcal{F}}(\mathcal{P}(d)), h'')) \wedge (h'' = (d.\mathrm{H} \parallel \text{"}A_1{:}d.\mathrm{A}_1,\ ...,\ A_n{:}d.\mathrm{A_n}\text{"}))\}$. □

Here, $\pi_{\mathcal{F}}(\mathcal{P}(d))$ works in the same way as projection in the classical relational algebra except it does not remove duplicates and it gracefully ignores fields in $\mathcal{F}$ which do not appear in $\mathcal{P}(d)$'s schema.

For example if $\mathcal{F}$ = "Data1" and if our primary key for ANN(r_3) was "Data1,Data2", then $ann''_r = \pi_{\mathcal{F}}(\mathrm{ANN}(r_3))$ will be

Data1	H	Day	Month	Year	L_t	U_t
D1	Data2:D2	2	8	1997	0.20	0.40
D1	Data2:D3	2	8	1997	0.30	0.40
D1	Data2:D3	3	8	1997	0.15	0.20

Notice that if we did not have the hidden field h'', then we would not be able to tell whether (D1) refers to event (D1,D2) or to event (D1,D3). In other words, the hidden field helps us to prevent loss of information. Now suppose that after a projection, we wanted (D1) to refer to all events where Data1 = D1. For the example above, this would mean that event (D1) should refer to the compound event ((D1,D2) ∨ (D1,D3)). This interpretation is not directly supported by our algebras because the disjunctive event above is *not instantaneous*.

To help reduce the size of the hidden field, projection only retains field-value pairs for fields which appear in the relation's primary key. Thus when the primary key is small, h'' will also be small.

5.8 Join of two annotated relations

For simplicity, this paper will only consider the "natural join" operation.

DEFINITION 5.11 (JOIN OF TWO ANNOTATED RELATIONS). Let selection condition $\mathcal{C}$ be defined as $(ann_r.\mathcal{L}_1 = ann'_r.\mathcal{L}_1) \wedge \ldots \wedge (ann_r.\mathcal{L}_n = ann'_r.\mathcal{L}_n))$ where "$\mathcal{L}_1 \ldots \mathcal{L}_n$" is the list of all manifest data fields which occur in the schema for both ann_r and ann'_r. Then the *join of annotated relations ann_r and ann'_r. under the α probabilistic conjunction strategy*, denoted $ann_r \bowtie_\alpha ann'_r$, is defined as $\pi_{\mathcal{F}}(\sigma_{\mathcal{C}}(ann_r \times_\alpha ann'_r)$ where $\mathcal{F}$ is the list of all manifest data fields which occur in the schema for either ann_r or ann'_r after removing duplicate field names. □

For example, $ann''_r = \mathrm{ANN}(r_3) \bowtie_{pc} \mathrm{ANN}(r_4)$ will be

Data1	Data2	H	Day	Month	Year	L_t	U_t
D1	D2		2	8	1997	0.10	0.40
D1	D3		3	8	1997	0.15	0.20

Notice that all of the hidden fields in ann''_r above are EMPTY. This occurs since in our example, $\mathcal{F}$ = "Data1,Data2" so when we perform a projection, the list of manifest data fields not appearing in $\mathcal{F}$ (i.e., the "$A_1, \ldots, A_n$" list in Definition 5.10) is empty.

Although our definition of join in this section only corresponds to a natural join, it can easily be extended to handle other types of join. For instance, an implementation which uses an SQL-like interface may allow users to explicitly specify appropriate values for $\mathcal{C}$ and $\mathcal{F}$.

5.9 Granularity of operations in TATA.

Complex expressions in our algebra can be used to perform operations involving a number of different combination functions. For instance, a user who wants to take a *join* of two annotated relations ann_{r_1} and ann_{r_2} assuming independence on the first day of each month and positive correlation on all other days may do so by writing a complex query. First, he selects from each relation, all first days of the month, leading to relations $ann_{r_1^1}$ and $ann_{r_2^1}$ respectively. He also selects from each of the two annotated relations ann_{r_1} and ann_{r_2} all tuples not dealing with the first day of the month, leading to relations $ann_{r_1^2}$ and $ann_{r_2^2}$ respectively. He now takes the join of $ann_{r_1^1}$ and $ann_{r_2^1}$ using independence, and the join of $ann_{r_1^2}$ and $ann_{r_2^2}$ using positive correlation, and then takes the multiset union of the two. Similarly, if different combination functions are desired for *compacting* different parts of a TP-relation, complex expressions in TATA can be used to specify that. For example, $\kappa_{\chi_1}(\sigma_{month=10}(r)) \cup \kappa_{\chi_2}(\sigma_{month\neq 10}(r))$ specifies that compaction for parts of r referring to the month of October should be performed under different combination strategy than the compaction for the rest of the relation.

In general, the person querying the data must - using his knowledge of the data - specify the combination strategies and/or dependencies he plans to use. However, the algebraic operations - as shown above - allow him to express queries taking into account, dependencies he has identified, even if different types of dependencies apply to different parts of the data. Similar comments apply to the TP-algebra defined below.

6. TP-ALGEBRA

This is the most important section of the paper. As mentioned several times before, for every data-tuple d and every time-point t, the TATA algebra *explicitly* represents the probability that a data-tuple d is in a given relation at time t. As pointed out by Dyreson and Snodgrass[11], this leads to a completely unacceptable explosion in the size of annotated relations and leads to major scalability problems. **In this section, we will show that TP-relations, which *implicitly* and *compactly* represent temporal probabilistic data, can be very efficiently manipulated by algebraic operations that correctly implement (as defined below) all the operations on the TP-algebra. In other words, we can use the TP-representation to efficiently implement operations analogous to the TATA algebra operations.**

In this section, we provide definitions for *TP-compression*, *compaction*, *intersection*, *union*, *selection*, *difference*, *cartesian product*, *projection* and *join* of TP-relations. With the exception of TP-compression, which has no analog in TATA, we will show that each of these operations *correctly implement* the corresponding operations in the TATA. The advantage is immediate: as TP-relations are relatively small when compared to their annotated counterparts, a huge savings, both in space (of storing TP- vs. annotated relations) and time (in terms of the time to process these operations) will result. Proofs of the correctness theorems in this section can be found in [9].

DEFINITION 6.1 (CORRECTLY IMPLEMENTS). Unary TPA operator op^T *correctly implements* the semantics for unary TATA operator op^A iff $\mathrm{ANN}(op^T(r)) = op^A(\mathrm{ANN}(r))$ for every TP-relation r.

Furthermore, binary TPA operator op^T *correctly implements* the semantics for binary TATA operator op^A iff $\mathrm{ANN}(r\, op^T\, r') = \mathrm{ANN}(r)\, op^A\, \mathrm{ANN}(r')$ for every pair of TP-relations r, r'. □

Note that as usual, intersection, union, and difference are only defined when both TP-relations have the same schema, selections are only defined when $\mathcal{C}$ is not an *inapplicable condition*, and projections are only defined when field list $\mathcal{F}$ is *projectable*.

The definitions in this section will produce a new TP-relation r'' based on input from consistent TP- relations r, r'. Oftentimes, these definitions will refer to TP-tuples tp, tp' which are assumed to be of the form $tp = (d, \gamma)$ and $tp' = (d', \gamma')$. Here, let γ contain n TP-cases of the form $\gamma_i = \langle C_i, D_i, L_i, U_i, \delta_i\rangle \in \gamma$ and let γ' contain n' TP-cases of the form $\gamma'_j = \langle C'_j, D'_j, L'_j, U'_j, \delta'_j\rangle \in \gamma'$.

6.1 TP-compression of a TP-relation

Let $N(r)$ and $N(tp)$ denote the number of TP-cases in TP-relation r and TP-tuple tp respectively. Intuitively, when we apply a *TP-compression function* to r, we may be able to decrease $N(r)$ and thus store temporal probabilistic information more compactly..

DEFINITION 6.2 (TP-COMPRESSION FUNCTION). A *TP-compression function* $\Xi(r)$ is a function which takes TP-relation r as input, and returns as output a TP-relation r'' where (i) $N(r'') \leq N(r)$ and (ii) there exists a bijection between $\mathrm{ANN}(r)$ and $\mathrm{ANN}(r'')$ which maps each $(d, t, L_t, U_t) \in \mathrm{ANN}(r)$ to a $(d, t, L_t, U''_t) \in \mathrm{ANN}(r'')$ such that $L_t \leq U''_t \leq U_t$. □

In other words, $\mathrm{ANN}(r'')$ and $\mathrm{ANN}(r)$ must have the same data tuples, time points, and lower bounds, but we allow TP-compressions to tighten upper bounds. Note that there are many functions which satisfy the definition of a TP-compression function given above. For instance, the following TP-compression functions combine TP-cases which share the same distribution and take advantage of the uniform distribution's regularity respectively.

DEFINITION 6.3 (TP-COMPRESSION OF A TP-RELATION; SAME-DISTRIBUTION). The *same-distribution TP-compression of TP-relation* r, denoted $\Xi^{sd}(r)$, is equal to the multiset S which can be constructed in the following way: Initially, let $S = r$. Then for each $(d, \gamma'') \in S$ and for each pair of TP-cases $\gamma_i = \langle C_i, D_i, L, U, \delta\rangle, \gamma_j = \langle C_j, D_j, L, U, \delta\rangle \in \gamma''$ where $\mathrm{sol}(D_i) = \mathrm{sol}(D_j)$, remove γ_i, γ_j from γ'' and add TP-case $\langle (C_i \vee C_j), D_i, L, U, \delta\rangle$ to γ''. □

Definition 6.4 (TP-compression of a TP-relation; u-based). The *u-based TP-compression of TP-relation* r, denoted $\Xi^u(r)$, is equal to the multiset S which can be constructed in the following way: Initially, let $S = r$. Then for each $(d, \gamma'') \in S$ and for each pair of TP-cases $\gamma_i = \langle C_i, D_i, L_i, U_i, u\rangle$, $\gamma_j = \langle C_j, D_j, L_j, U_j, u\rangle \in \gamma''$ where $n_i = |\mathrm{sol}(D_i)|$, $n_j = |\mathrm{sol}(D_j)|$, and $([L_t, U_t] = \frac{1}{n_i} \cdot [L_i, U_i] = \frac{1}{n_j} \cdot [L_j, U_j])$, remove γ_i, γ_j from γ'' and add TP-case $\langle (C_i \vee C_j), (D_i \vee D_j), L_{ij}, \min(1, U_{ij}), u\rangle$ to γ'' where $n_{ij} = |\mathrm{sol}(D_i \vee D_j)|$ and $[L_{ij}, U_{ij}] = n_{ij} \cdot [L_t, U_t]$. □

Note that when $U_{ij} > 1$, the upper bounds in $\mathrm{ANN}(r'')$ will be tighter than the ones in $\mathrm{ANN}(r)$.

Definition 6.5 (TP-compression of a TP-relation; hybrid). The *hybrid TP-compression of TP-relation* r, denoted $\Xi^{hy}(r)$, is defined as $\Xi^{sd}(\Xi^u(r))$. □

The following theorem indicates that the functions above satisfy our definition for a TP-compression.

Theorem 6. *$\Xi^{sd}(r)$, $\Xi^u(r)$ and $\Xi^{hy}(r)$ are all TP-compression functions.*

More sophisticated TP-compression operators are also possible. For instance, let $p \in (0,1)$ be a probability and let $t_1, \ldots, t_n$ be a list of (consecutive) time points in S_τ where for each $1 \le i < n$, $t_{i+1} = next_\tau(t_i)$. Then if γ'' contains n TP-cases of the form $\langle (\#), (t_i), L''_{t_i}, U''_{t_i}, u\rangle$ where $([L''_{t_{i+1}}, U''_{t_{i+1}}] = p \cdot [L''_{t_i}, U''_{t_i}]$ for all $1 \le i < n)$ and $(U''_{t_1} \div p \le 1)$, apply an operator which replaces these n TP-cases with the TP-case $\langle (\#), (t_1 \sim t_n), L'', U'', \delta''\rangle$ where $[L'', U''] = [L''_{t_1} \div p, U''_{t_1} \div p]$ and $\delta'' =$ "g, p".

The aforementioned operator performs a *g-based TP-compression*. Note that for any distribution function δ, one can define a corresponding δ-based TP-compression operator. Also note that one can obtain optimal TP-compression (i.e., the smallest possible value for $N(r''))$ by allowing a TP-compression operator to dynamically create new distribution functions which fit the resulting data.

6.2 Compaction of a TP-relation

As in the case of the TATA, the *compaction* operation in the TPA will be used to define the operations of *intersection*, *union* and *projection*.

Definition 6.6 (Compaction of a TP-relation). A function κ from TP-relations to TP-relations is called a *compaction operation* if it satisfies the following axioms:

- Compactness : $\kappa(r)$ is *compact* for all TP-relations r.
- No Fooling Around (NFA) : If r is compact then $\mathrm{ANN}(\kappa(r)) = \mathrm{ANN}(r)$.
- Conservativeness : If $at = (d, t, L_t, U_t) \in \mathrm{ANN}(\kappa(r))$, then $\exists at' = (d, t, L'_t, U'_t) \in \mathrm{ANN}(r)$. □

The Compactness, NFA, and Conservativeness axioms for TP-relations are similar in spirit and intuition to the same concepts defined earlier for annotated relations. However, one difference is that while each annotated relation has a unique compaction, given an annotated relation ann_r, there may be more than one TP-relation r such that $ann_r = ANN(r)$. Hence, there may be more than one TP-relation r' which is a compaction of r. *Any* such tp-relation r'' is a valid result of compaction.

We now develop a *compaction algorithm*, which constructs a compaction of a TP-relation.

As in the case of TATA, it is possible (using the selection operator) to (i) split a relation into 2 or more parts, (ii) compact each part using a local (to that part) combination function, and (iii) take the unions of the results. Thus, compactions can be performed by applying different combination functions to different parts of a relation.

As in the case of TATA, there are many different compaction operations on TP-relations. Below, we present the TP analogs of χ-compactions and p-strategy based compactions.

DEFINITION 6.7 (χ-COMPACTION OF A TP-RELATION). Let χ be a combination function. Then the
χ-compaction of TP-relation r, is any relation $\kappa_\chi(r)$ such that $\mathrm{ANN}(\kappa_\chi(r)) =$
$\{at = (d,t,L_t,U_t) \mid [L_t,U_t] = \chi(\{[L_1^{(d,t)},U_1^{(d,t)}],\ldots,[L_k^{(d,t)},U_k^{(d,t)}]\})\}$ where
$\mathrm{ANN}(r)[d,t] = \{at_1^{(d,t)},\ldots,at_k^{(d,t)}\}$ and $at_i^{(d,t)} = (d,t,L_i^{(d,t)},U_i^{(d,t)})$. □

The following lemma states that this operation is indeed a *compaction* operation.

LEMMA 1. *Let χ be a combination function. Then $\kappa_\chi(r)$ is a compaction operation.*

Algorithm **Compute-Compaction** shown below provides a mechanism to efficiently compute compactions without resorting to annotation. This algorithm can perform compactions using either a combination function χ or a p-strategy ρ. The boxed line in this algorithm shows exactly where a combination function or p-strategy is applied to compact data-identical tuples.

THEOREM 7. *Let χ be a combination function. Then algorithm* **Compute-Compaction**(r,χ) *correctly computes the $\kappa_\chi(r)$ compaction operation.*

We define p-strategy based compactions of TP-relations in the same way as $\kappa_\chi(r)$ except we let $[L_t,U_t] = ([L_1^{(d,t)},U_1^{(d,t)}] \otimes \ldots \otimes [L_k^{(d,t)},U_k^{(d,t)}])$ and let $[L_t,U_t] = ([L_1^{(d,t)},U_1^{(d,t)}] \oplus \ldots \oplus [L_k^{(d,t)},U_k^{(d,t)}])$ when defining $\kappa_\otimes(r)$ and $\kappa_\oplus(r)$ respectively.

LEMMA 2. *Let ρ be a p-strategy. Then $\kappa_\rho(r)$ is a compaction operation.*

As p-strategy based compaction of TP-relations is defined declaratively, we need an explicit algorithm (mentioned above) to compute it. The following result states the correctness of this algorithm.

THEOREM 8. *Let ρ be a (conjunctive or disjunctive) p-strategy. Then algorithm* **Compute-Compaction**(r,ρ) *correctly computes the $\kappa_\rho(r)$ compaction operation.*

Thus far, we have separately defined compaction operators on annotated relations and on TP-relations. The following definition specifies when a compaction operator on the annotated side corresponds to a compaction operator on the TP-side.

DEFINITION 6.8 (COMPATIBLE PAIR OF COMPACTIONS). A pair $\langle \kappa^A(.), \kappa^T(.)\rangle$ of compaction operators is a *compatible pair* iff for every TP-relation r, $\kappa^A(\mathrm{ANN}(r)) = \mathrm{ANN}(\kappa^T(r))$, i.e., iff $\kappa^A \circ ANN = \kappa^T$. □

Algorithm Compute-Compaction(r,f):
Input: TP-relation r and combination function or p-strategy f
Output: TP-relation $r'' = \kappa_f(r)$
$r'' := \emptyset$; `// Initialize the resulting relation`
$r' := r$; `// Obtain a working copy of the initial relation`
`// For each (maximal) multiset S of data-identical TP-tuples in r`
while $(r' \neq \emptyset)$ **do** {
  Select a TP-tuple $tp \in r'$;
  $S := r'[tp]$; `// Extract the next equivalence class from r'`
  $r' := r' - S$; $\gamma'' := \emptyset$;
  $\Gamma_S := \biguplus_{(d,\gamma)\in S} \gamma$; $\Gamma_I := \Gamma_S$;
  foreach $\gamma_i = \langle C_i, D_i, L_i, U_i, \delta_i\rangle \in \Gamma_I$ {
    Remove γ_i from Γ_I and Γ_S; $C_S := \bigvee_{\langle C,D,L,U,\delta\rangle\in\Gamma_S} C$;
    if $\mathrm{sol}(C_i \wedge \neg C_s) \neq \emptyset$ **then** Add $\langle (C_i \wedge \neg C_s), D_i, L_i, U_i, \delta_i\rangle$ to γ'';
    if $\mathrm{sol}(C_i \wedge C_s) \neq \emptyset$ **then** Add $\langle (C_i \wedge C_s), D_i, L_i, U_i, \delta_i\rangle$ to Γ_S; }
  $C_S := \bigvee_{\langle C,D,L,U,\delta\rangle\in\Gamma_S} C$;
  `// Note: Each` $t \in \mathrm{sol}(C_S)$ `will refer to more than one TP-case`
  foreach $t \in \mathrm{sol}(C_S)$ {
    $X := \emptyset$; `//` X `will contain the probability intervals to be combined`
    $\Gamma_t := \{\langle C, D, L, U, \delta\rangle \in \Gamma_S \mid t \in \mathrm{sol}(C)\}$;
    foreach $\langle C, D, L, U, \delta\rangle \in \Gamma_t$ {
      $x_t := \delta(D, t)$; $[L_t, U_t] := [L \cdot x_t, U \cdot x_t]$;
      $X := X \cup \{[L_t, U_t]\}$; }
    $\boxed{[L''_t, U''_t] := f(X);}$
    Add TP-case $\langle (\#), (t), L''_t, U''_t, u\rangle$ to γ''; }
  Add TP-tuple (d, γ'') to r''; }
return r'';
End-Algorithm

The following two theorems say that for any arbitrary combination function χ and for any arbitrary p-strategy ρ, $\langle \kappa_\chi(\mathrm{ANN}(r)), \kappa_\chi(r)\rangle$ and $\langle \kappa_\rho(\mathrm{ANN}(r)), \kappa_\rho(r)\rangle$ are compatible pairs.

THEOREM 9. *Let χ be any combination function. Then $\langle \kappa_\chi(\mathrm{ANN}(r)), \kappa_\chi(r)\rangle$ is a compatible pair.*

THEOREM 10. *Let ρ be any p-strategy. Then $\langle \kappa_\rho(\mathrm{ANN}(r)), \kappa_\rho(r)\rangle$ is a compatible pair.*

6.3 Intersection of two TP-relations

In this section, we show how we can *correctly implement* the intersection of two TP-relations. Intersection consists of of two suboperations — *multiset intersection* and *combination function based compaction.*

DEFINITION 6.9 (MULTISET INTERSECTION OF TWO TP-RELATIONS). The *multiset intersection of TP-relations r and r'*, denoted $r \cap r'$, can be constructed in the following way: Initially, let $r'' = \emptyset$. Then for each $tp = (d, \gamma) \in r$ and each $tp' = (d', \gamma') \in r'$ where $(d = d')$,
1. Let $\Gamma = \Gamma' = \emptyset$.
2. For each $\gamma_i \in \gamma$ and each $\gamma'_j \in \gamma'$ where $|\mathrm{sol}(C_i \wedge C'_j)| \geq 1$, add TP-case

$\langle (C_i \wedge C'_j), D_i, L_i, U_i, \delta_i \rangle$ to Γ and add TP-case $\langle (C_i \wedge C'_j), D'_j, L'_j, U'_j, \delta'_j \rangle$ to Γ'. Note that $(C_i \wedge C'_j)$ is shared by both TP-cases.
3. If $\Gamma \neq \emptyset$, add TP-tuples (d, Γ) and (d, Γ') to r''. Note that $\Gamma \neq \emptyset \Rightarrow \Gamma' \neq \emptyset$. Γ will be empty if there are no overlapping time points. □

Definition 6.10 (intersection of two TP-relations). The *intersection of TP-relations r and r' under the χ combination function*, denoted $r \cap_\chi r'$, is defined as $\kappa_\chi(r \cap r')$. □

As in the case of the TATA, we apply a κ_χ compaction operator to the result of a multiset intersection. Also, to keep the size of $r \cap_\chi r'$ manageable, we usually perform a TP-compression on the result of a compaction.
For example, $r'' = r_1 \cap r_2$ will be

Data	H	C	D	L	U	δ
D1		(2/8/1997 ∼ 3/8/1997)	(1/8/1997 ∼ 3/8/1997)	0.64	0.88	g
		(6/8/1997 ∼ 8/8/1997)	(5/8/1997 ∼ 8/8/1997)	0.40	0.80	u
D1		(2/8/1997 ∼ 3/8/1997)	(2/8/1997 ∼ 3/8/1997)	0.20	0.50	g
		(6/8/1997 ∼ 8/8/1997)	(6/8/1997 ∼ 9/8/1997)	0.40	0.80	u

and, $\Xi^{hy}(r \cap_{eq} r') = \Xi^{hy}(\kappa_{eq}(r''))$ will be

Data	H	C	D	L	U	δ
D1		(#)	(2/8/1997 ∼ 2/8/1997)	0.16	0.22	u
		(#)	(3/8/1997 ∼ 3/8/1997)	0.08	0.11	u
		(#)	(6/8/1997 ∼ 8/8/1997)	0.30	0.60	u

The following shows that our definition of intersection correctly implements the TATA semantics. This lets us completely avoid the construction of the (huge) annotated expansion while preserving the same semantics.

Theorem 11 (Correctness of intersection). $\mathrm{ANN}(r \cap_\chi r') = \mathrm{ANN}(r) \cap_\chi \mathrm{ANN}(r')$.

6.4 Union of two TP-relations

In this section, we show how we can *correctly implement* the union of two TP-relations. Union consists of two suboperations – *multiset union* and *combination function based compaction.*

Definition 6.11 (multiset union of two TP-relations). The *multiset union of TP-relations r and r'*, denoted $r \cup r'$, is defined as $r'' = r \uplus r'$. □

Intuitively, r'' will contain all $tp \in r$ and all $tp' \in r'$. For example, $r'' = r_1 \cup r_2$ will be

Data	H	C	D	L	U	δ
D1		(#)	(1/8/1997 ∼ 3/8/1997)	0.64	0.88	g
		(#)	(5/8/1997 ∼ 8/8/1997)	0.40	0.80	u
D1		(#)	(2/8/1997 ∼ 3/8/1997)	0.20	0.50	g
		(#)	(6/8/1997 ∼ 9/8/1997)	0.40	0.80	u

As in the case of the TATA, we apply a κ_χ compaction operator to the result of a multiset union.

DEFINITION 6.12 (UNION OF TWO TP-RELATIONS). The *union of TP-relations r and r' under the χ combination function*, denoted $r \cup_\chi r'$, is defined as $\kappa_\chi(r \cup r')$. □

For example, $\Xi^{hy}(r \cup_{eq} r') = \Xi^{hy}(\kappa_{eq}(r''))$ will be

Data	H	C	D	L	U	δ
D1		(1/8/1997)	(1/8/1997 $\sim$ 3/8/1997)	0.64	0.88	g
		(#)	(2/8/1997 $\sim$ 2/8/1997)	0.16	0.22	u
		(#)	(3/8/1997 $\sim$ 3/8/1997)	0.08	0.11	u
		(#)	(5/8/1997 $\sim$ 9/8/1997)	0.50	1.00	u

The following shows that our definition of union correctly implements the TATA semantics.

THEOREM 12 (CORRECTNESS OF UNION). $\mathrm{ANN}(r \cup_\chi r') = \mathrm{ANN}(r) \cup_\chi \mathrm{ANN}(r')$.

6.5 Selection on a TP-relation

In this section, we show how we can *correctly implement* selection on a TP-relation. The *TP-filter* operator defined below will help us handle selections of *probabilistic conditions* (§5.4) on TP-relations.

DEFINITION 6.13 (TP-FILTER). Let $\gamma_i = \langle C_i, D_i, L_i, U_i, \delta_i \rangle$ be a TP-case, let $\mathcal{C} = (F \; op \; v)$ be a probabilistic condition, and let $x = L_i$ if $F =$ "L" or let $x = U_i$ otherwise. Then a *TP-filter* is a function which takes γ_i and $\mathcal{C}$ as input, and returns as output a temporal constraint C''_i where
1. $\mathrm{sol}(C''_i) \subseteq \mathrm{sol}(C_i)$
2. For each time point $t \in \mathrm{sol}(C''_i)$, $(x_t \; op \; v)$ must be true when $x_t = \delta_i(D_i, t) \cdot x$
3. There is no temporal constraint C'_i where $(\mathrm{sol}(C'_i) \supset \mathrm{sol}(C''_i))$ and C'_i satisfies the previous cases.

Intuitively, a TP-filter returns a temporal constraint whose solution set consists of all time points $t \in \mathrm{sol}(C_i)$ where $[L_t, U_t] = [L_i \cdot \delta_i(D_i, t), U_i \cdot \delta_i(D_i, t)]$ satisfies $\mathcal{C}$. If no $t \in \mathrm{sol}(C_i)$ satisfies this condition, TP-filter$(\gamma_i, \mathcal{C})$ returns an inconsistent temporal constraint. □

For example if $\gamma_i = \langle (\#), (5/8/1997 \sim 8/8/1997), 0.4, 0.8, g \rangle$ and $\mathcal{C} = (U > 0.15)$, TP-filter$(\gamma_i, \mathcal{C})$ will be $C''_i = (5/8/1997 \sim 6/8/1997)$ since $(0.4 > 0.15)$ for 5/8/1997 and $(0.2 > 0.15)$ for 6/8/1997.

In general, $n = |\mathrm{sol}(C_i)|$ may be a large number. With arbitrary distribution functions, this can be problematic since the TP-filter function may have to test all n time points. Fortunately, this problem can be alleviated by exploiting regularities in our distribution functions. For instance if $\delta_i =$ "u", then we only need to test one time point $t \in \mathrm{sol}(C_i)$; if t should be in $\mathrm{sol}(C''_i)$, then $C''_i = C_i$ or $C''_i = \emptyset$ otherwise. This "all or none" behavior occurs since each $t \in \mathrm{sol}(C_i)$ will have the same probability value after distributing uniformly.

Implementations of TP-filters can also exploit regularities in the geometric PDF by searching $\mathrm{sol}(C_i)$ in chronological (or reverse chronological) order and then ending the search after finding the first t which should not be in $\mathrm{sol}(C''_i)$. The exact search method to use will, of course, depend on which *op* is present in $\mathcal{C}$. For instance if $op = (\neq)$, it may be cheaper to let $C'_i =$ TP-filter$(\gamma_i, \neg\mathcal{C})$ and then return $C''_i = (C_i \wedge \neg C'_i)$.

Definition 6.14 (Selection on a TP-tuple; atomic condition). The *selection of atomic condition $\mathcal{C}$ on TP-tuple $tp = (d, \gamma)$*, denoted $\sigma_{\mathcal{C}}(tp)$, can be constructed in the following way: Initially, let $\gamma'' = \emptyset$.
• If $\mathcal{C}$ is a data condition, let $\gamma'' = \gamma$ if d satisfies $\mathcal{C}$.
• If $\mathcal{C}$ is a temporal condition, then for each $\gamma_i \in \gamma$ where $C''_i = (C_i \wedge \mathcal{C})$ is consistent, add TP-case $\langle C''_i, D_i, L_i, U_i, \delta_i \rangle$ to γ''.
• If $\mathcal{C}$ is a probabilistic condition, then for each $\gamma_i \in \gamma$ where $C''_i = \text{TP-filter}(\gamma_i, \mathcal{C})$ is consistent, add TP-case $\langle C''_i, D_i, L_i, U_i, \delta_i \rangle$ to γ''.
If $\gamma'' = \emptyset$ then $\sigma_{\mathcal{C}}(tp) = \emptyset$. Otherwise, $\sigma_{\mathcal{C}}(tp) = (d, \gamma'')$. □

Definition 6.15 (Selection on a TP-relation; atomic condition). The *selection of atomic condition $\mathcal{C}$ on TP-relation $r = \{tp_1, \ldots, tp_n\}$*, denoted $\sigma_{\mathcal{C}}(r)$, is defined as $(\sigma_{\mathcal{C}}(tp_1) \uplus \ldots \uplus \sigma_{\mathcal{C}}(tp_n))$. □

Note that for all $tp_i, tp_j \in r$, $\sigma_{\mathcal{C}}(tp_i)$ does not affect the results of $\sigma_{\mathcal{C}}(tp_j)$ when computing $\sigma_{\mathcal{C}}(r)$.

For example if $\mathcal{C} = (2/8/1997 \sim 7/8/1997)$, $\sigma_{\mathcal{C}}(r_1)$ will be

Data	H	C	D	L	U	δ
D1		$(2/8/1997 \sim 3/8/1997)$	$(1/8/1997 \sim 3/8/1997)$	0.64	0.88	g
		$(5/8/1997 \sim 7/8/1997)$	$(5/8/1997 \sim 8/8/1997)$	0.40	0.80	u

and $\sigma_{\mathcal{C}}(r_2)$ will be

Data	H	C	D	L	U	δ
D1		$(2/8/1997 \sim 3/8/1997)$	$(2/8/1997 \sim 3/8/1997)$	0.20	0.50	g
		$(6/8/1997 \sim 7/8/1997)$	$(6/8/1997 \sim 9/8/1997)$	0.40	0.80	u

but if $\mathcal{C} = (L \neq 0.10)$, $\sigma_{\mathcal{C}}(r_1)$ will be

Data	H	C	D	L	U	δ
D1		$(\#)$	$(1/8/1997 \sim 3/8/1997)$	0.64	0.88	g

and $\sigma_{\mathcal{C}}(r_2)$ will be

Data	H	C	D	L	U	δ
D1		$(3/8/1997)$	$(2/8/1997 \sim 3/8/1997)$	0.20	0.50	g

We can extend selection to handle non-atomic selection conditions by using the following definition.

Definition 6.16 (Selection on a TP-relation). The *selection of condition $\mathcal{C}$ on TP-relation r*, denoted $\sigma_{\mathcal{C}}(r)$, is defined inductively in the following way:
• If $\mathcal{C}$ is an atomic condition, then $r'' = \sigma_{\mathcal{C}}(r)$ by way of our previous definition.
• If $\mathcal{C}$ is of the form $(\mathcal{C}_1 \wedge \mathcal{C}_2)$, then $r'' = \sigma_{\mathcal{C}_1}(\sigma_{\mathcal{C}_2}(r))$.
• If $\mathcal{C}$ is of the form $(\mathcal{C}_1 \vee \mathcal{C}_2)$, then $r'' = \sigma_{\mathcal{C}_1}(r) \cup_{eq} \sigma_{\mathcal{C}_2}(r)$. (Note that as long as r is compact, it follows by the Identity axiom that irrespective of which combination function is used, we obtain the same results, i.e. "eq" in the above definition can be replaced by any other combination without the result being changed.) • If $\mathcal{C}$ is of the form $(\neg \mathcal{C}_1)$, then

- ⋆ If $\mathcal{C}_1$ is of the form $(\mathcal{C}_2 \wedge \mathcal{C}_3)$, $(\mathcal{C}_2 \vee \mathcal{C}_3)$, or $(\neg \mathcal{C}_2)$, then $r'' = \sigma_{\mathcal{C}_4}(r)$ where $\mathcal{C}_4 = (\neg \mathcal{C}_2 \vee \neg \mathcal{C}_3)$, $\mathcal{C}_4 = (\neg \mathcal{C}_2 \wedge \neg \mathcal{C}_3)$, or $\mathcal{C}_4 = (\mathcal{C}_2)$ respectively.
- ⋆ If $\mathcal{C}_1$ is a *data*, *temporal*, or *probabilistic* condition, then $r'' = \sigma_{\mathcal{C}_4}(r)$ where $\mathcal{C}_4$ is the atomic, logical negation of $\mathcal{C}_1$.

⋆ Otherwise, $\mathcal{C}_1$ is an *inapplicable* condition and so r'' is not defined.

To perform selections with non-atomic conditions on annotated relations, use the abiove definition except replace all instances of r and r'' with ANN(r) and ANN(r'') respectively. □

For example if $\mathcal{C}_1 = (2/8/1997 \sim 7/8/1997)$, $\mathcal{C}_2 = (L \neq 0.10)$, and $\mathcal{C} = (\mathcal{C}_1 \wedge \mathcal{C}_2)$, then $\sigma_{\mathcal{C}}(r_1)$ will be

Data	H	C	D	L	U	δ
D1		$(2/8/1997 \sim 3/8/1997)$	$(1/8/1997 \sim 3/8/1997)$	0.64	0.88	g

and $\sigma_{\mathcal{C}}(r_2)$ will be

Data	H	C	D	L	U	δ
D1		$(3/8/1997)$	$(2/8/1997 \sim 3/8/1997)$	0.20	0.50	g

Theorem 13. $\sigma_{\mathcal{C}_1}(\sigma_{\mathcal{C}_2}(r)) = \sigma_{\mathcal{C}_2}(\sigma_{\mathcal{C}_1}(r))$.

The following table shows how one may generate queries (on a TP-relation r) which correspond to seven of J. F. Allen's thirteen possible temporal relationships [1]. The six remaining possibilities correspond to the inverses of these original seven (the inverse of "equal" is identical to "equal" so it is not counted). Here, we ensure that r uses two TP-tuples for each continuous duration event e. Specifically, suppose e can be described by data tuple d with relational schema A. Then r is a TP-relation over relational schema (A,Kind) where $dom(Kind) = \{S, E\}$. r contains a TP-tuple (d,"S",γ_S) for instantaneous event $st(e)$, and a TP-tuple (d,"E",γ_E) for instantaneous event $end(e)$.

Description	Specification	Query	Conditions
e before e_q	$end(e) \leq st(e_q)$	$\sigma_{\mathcal{C}}(r)$	$\mathcal{C} = ((t_S \sim t_1) \wedge (\text{Kind} = E))$
e equal e_q	$st(e) = st(e_q) \wedge$ $end(e) = end(e_q)$	$\sigma_{\mathcal{C}_1}(r) \cap_{eq} \sigma_{\mathcal{C}_2}(r)$	$\mathcal{C}_1 = ((t_1 \sim t_1) \wedge (\text{Kind} = S))$ $\mathcal{C}_2 = ((t_2 \sim t_2) \wedge (\text{Kind} = E))$
e meets e_q	$end(e) = st(e_q)$	$\sigma_{\mathcal{C}}(r)$	$\mathcal{C} = ((t_1 \sim t_1) \wedge (\text{Kind} = E))$
e overlaps e_q	$st(e) \leq st(e_q) \wedge$ $st(e_q) \leq end(e)$	$\sigma_{\mathcal{C}_1}(r) \cap_{eq} \sigma_{\mathcal{C}_2}(r)$	$\mathcal{C}_1 = ((t_S \sim t_1) \wedge (\text{Kind} = S))$ $\mathcal{C}_2 = ((t_1 \sim t_E) \wedge (\text{Kind} = E))$
e during e_q	$st(e_q) \leq st(e) \wedge$ $end(e) \leq end(e_q)$	$\sigma_{\mathcal{C}_1}(r) \cap_{eq} \sigma_{\mathcal{C}_2}(r)$	$\mathcal{C}_1 = ((t_1 \sim t_E) \wedge (\text{Kind} = S))$ $\mathcal{C}_2 = ((t_S \sim t_2) \wedge (\text{Kind} = E))$
e starts e_q	$st(e) = st(e_q) \wedge$ $end(e) \leq end(e_q)$	$\sigma_{\mathcal{C}_1}(r) \cap_{eq} \sigma_{\mathcal{C}_2}(r)$	$\mathcal{C}_1 = ((t_1 \sim t_1) \wedge (\text{Kind} = S))$ $\mathcal{C}_2 = ((t_S \sim t_2) \wedge (\text{Kind} = E))$
e finishes e_q	$st(e_q) \leq st(e)$ $end(e) = end(e_q)$	$\sigma_{\mathcal{C}_1}(r) \cap_{eq} \sigma_{\mathcal{C}_2}(r)$	$\mathcal{C}_1 = ((t_1 \sim t_E) \wedge (\text{Kind} = S))$ $\mathcal{C}_2 = ((t_2 \sim t_2) \wedge (\text{Kind} = E))$

When executing these temporal queries, use the following rule: If $st(e)$ or $end(e)$ satisfies the selection condition, then the TP-tuples for both $st(e)$ and $end(e)$ should be returned. Thus, our queries will return TP-tuples for every event e which satisfies some relationship w.r.t. query event e_q where $st(e_q) = t_1$ and $end(e_q) = t_2$. Recall that t_S (t_E) denotes the earliest (latest) time point of a calendar.

Theorem 14 (Correctness of selection). $\text{ANN}(\sigma_{\mathcal{C}}(r)) = \sigma_{\mathcal{C}}(\text{ANN}(r))$.

6.6 Difference of two TP-relations

In this section, we show how we can *correctly implement* the difference of two TP-relations.

Definition 6.17 (difference of two TP-relations). The *difference of TP-relations r and r'*, denoted $r - r'$, can be constructed in the following way: Initially, let $r'' = r$. Then for each $tp = (d, \gamma) \in r''$ and each $tp' = (d', \gamma') \in r'$ where $(d = d')$,
1. Let $\gamma'' = \emptyset$ and let $C' = (C'_1 \vee \ldots \vee C'_{n'})$. Recall that tp' contains exactly n' TP-cases.
2. For each $\gamma_i \in \gamma$ where $C''_i = (C_i \wedge \neg C')$ is consistent, add TP-case $\langle C''_i, D_i, L_i, U_i, \delta_i \rangle$ to γ''.
3.Remove tp from r''. Then if $\gamma'' \neq \emptyset$, add TP-tuple (d, γ'') to r''. □

For example $r_1 - r_2$ and $r_2 - r_1$ will be

Data	H	C	D	L	U	δ
D1		(1/8/1997)	(1/8/1997 ∼ 3/8/1997)	0.64	0.88	g
		(5/8/1997)	(5/8/1997 ∼ 8/8/1997)	0.40	0.80	u

and

Data	H	C	D	L	U	δ
D1		(9/8/1997)	(6/8/1997 ∼ 9/8/1997)	0.40	0.80	u

Theorem 15 (Correctness of difference). $\text{ANN}(r - r') = \text{ANN}(r) - \text{ANN}(r')$.

6.7 Cartesian product of two TP-relations

As in the annotated case, that when taking the cartesian product of two relations, we must know the relationship between the events denoted by the tuples in the two relations because the the result of the Cartesian Product will contain the probability of their conjunction. Thus, conjunction strategies *parameterize* the Cartesian Product operation.

Definition 6.18 (cartesian product of two TP-relations). The *cartesian product of TP-relations r and r' under the α probabilistic conjunction strategy*, denoted $r \times_\alpha r'$, can be constructed in the following way: Initially, let $r'' = \emptyset$. Then for each $tp = (d, \gamma) \in r$ and each $tp' = (d', \gamma') \in r'$,
1. Let $\gamma'' = \emptyset$.
2. For each time point t where $t \in \text{sol}(C_i)$ for some $\gamma_i \in \gamma$ and $t \in \text{sol}(C'_j)$ for some $\gamma'_j \in \gamma'$,
   (a) Let $[L_t, U_t] = [L_i \cdot x_t, U_i \cdot x_t]$ where $x_t = \delta_i(D_i, t)$.
   (b) Let $[L'_t, U'_t] = [L'_j \cdot x'_t, U'_j \cdot x'_t]$ where $x'_t = \delta'_j(D'_j, t)$.
   (c) Let $[L''_t, U''_t] = ([L_t, U_t] \otimes_\alpha [L'_t, U'_t])$.
   (d) Add TP-case $\langle (\#), (t), L''_t, U''_t, u \rangle$ to γ''.
3. If $\gamma'' \neq \emptyset$, add TP-tuple (d'', γ'') to r'' where $d'' = (\mathcal{P}(d), \mathcal{P}(d'), h'')$ and $h'' = (d.\text{H} \parallel d'.\text{H})$. □

For example, $\Xi^{hy}(r_1 \times_{ig} r_2)$ will be

r_1.Data	r_2.Data	H	C	D	L	U	δ
D1	D1		(#)	(2/8/1997 ∼ 2/8/1997)	0.00	0.22	u
			(#)	(3/8/1997 ∼ 3/8/1997)	0.00	0.11	u
			(#)	(6/8/1997 ∼ 8/8/1997)	0.00	0.60	u

but $\Xi^{hy}(r_1 \times_{pc} r_2)$ will be

r_1.Data	r_2.Data	H	C	D	L	U	δ
D1	D1		(#)	(2/8/1997 ∼ 2/8/1997)	0.10	0.22	u
			(#)	(3/8/1997 ∼ 3/8/1997)	0.05	0.11	u
			(#)	(6/8/1997 ∼ 8/8/1997)	0.30	0.60	u

The use of a TP-compression operation when executing a Cartesian product operation is important because sometimes, Cartesian product can produce a large number of TP-cases when an existing tp-case gets broken into "pieces." TP-compressions prevent this from happening. The following shows that our definition of cartesian product correctly implements the TATA semantics.

Theorem 16 (Correctness of cartesian product). $\text{ANN}(r \times_\alpha r') = \text{ANN}(r) \times_\alpha \text{ANN}(r')$.

6.8 Projection on a TP-relation

In this section, we show how we can *correctly implement* projection on a TP-relation.

Definition 6.19 (projection on a TP-relation). Let $\mathcal{F}$ be a list of fields which are *projectable* w.r.t. r and let "$A_1, \ldots, A_n$" be the (possibly empty) list of all manifest data fields which appear in the primary key of r but do not appear in $\mathcal{F}$. Then the *projection of field list $\mathcal{F}$ on TP-relation r*, denoted $\pi_{\mathcal{F}}(r)$, can be constructed in the following way: Initially, let $r'' = \emptyset$. Then for each $(d, \gamma) \in r$, add TP-tuple (d'', γ) to r'' where $d'' = (\pi_{\mathcal{F}}(\mathcal{P}(d)), h'')$ and $h'' = (d.\text{H} \parallel$ "A_1:$d.\text{A}_1$, ..., A_n:$d.\text{A}_\text{n}$"). (Recall that the $\pi_{\mathcal{F}}(\mathcal{P}(d))$ operator was defined in section 5.7.) □

For example if $\mathcal{F}$ = "Data1" and our primary key for r_3 was "Data1,Data2", $r'' = \pi_{\mathcal{F}}(r_3)$ will be

Data1	H	C	D	L	U	δ
D1	Data2:D2	(#)	(2/8/1997 ∼ 2/8/1997)	0.20	0.40	u
D1	Data2:D3	(#)	(2/8/1997 ∼ 3/8/1997)	0.60	0.80	g

Theorem 17 (Correctness of projection). $\text{ANN}(\pi_{\mathcal{F}}(r)) = \pi_{\mathcal{F}}(\text{ANN}(r))$.

6.9 Join of two TP-relations

In this section, we show how we can *correctly implement* the join of two TP-relations.

Definition 6.20 (join of two TP-relations). Let selection condition $\mathcal{C}$ be $((r.\mathcal{L}_1 = r'.\mathcal{L}_1) \wedge \ldots \wedge (r.\mathcal{L}_n = r'.\mathcal{L}_n))$ where "$\mathcal{L}_1 \ldots \mathcal{L}_n$" is the list of all manifest data fields which occur in the schema for both r and r'. Then the *join of TP-relations r and r' under the α probabilistic conjunction strategy*, denoted $r \bowtie_\alpha r'$, is defined as $\pi_{\mathcal{F}}(\sigma_{C}(r \times_{\alpha} r'))$ where $\mathcal{F}$ is the list of all manifest data fields which occur in the schema for either r or r' after removing duplicate field names. □

For example, $r_3 \bowtie_{pc} r_4$ will be

Data1	Data2	H	C	D	L	U	δ
D1	D2		(#)	(2/8/1997)	0.10	0.40	u
D1	D3		(#)	(3/8/1997)	0.15	0.20	u

Theorem 18 (Correctness of join). *$r \bowtie_{\alpha} r' = \mathrm{ANN}(r) \bowtie_{\alpha} \mathrm{ANN}(r')$.*

7. IMPLEMENTATION AND EXPERIMENTS

All TPA operators described in this paper have been implemented under Borland C++ version 5.01. Our code can run on any 32 bit Windows platform (i.e., Win95, Win98, and WinNT). This code communicates with standard, relational databases by using the Borland Database Engine's API (BDE version 3.0). Here, the same API can be used to interface with a variety of databases including Paradox, dBASE, Oracle, Microsoft SQL Server, InterBase, Sybase, and any ODBC (Open Database Connectivity) data source. Note however that the underlying, relational database should (i) be capable of storing 32 bit integers and (ii) be able to process basic SQL queries. A demonstration of this implementation can be accessed from the web by clicking on the "TP-Databases" link in the "`http://bester.cs.umd.edu`" page — our user interface is fully compatible with the Internet Explorer 4.0 browser.

7.1 Experiments

We conducted two sets of experiments. The first set studies the relative efficiency of TP-algebra operations when compared to TATA algebra operations. In addition, this set of experiments was designed to study how different distribution functions affected the efficiency of operations. The second set of experiments tested scalability of the TP-algebra operations. The TATA algebra was implemented for these experiments by forcing TP-tuples to have only one TP-case where $C = D$, $|\mathrm{sol}(D)| = 1$, and $\delta =$"u" (uniform distribution).

All our experiments were conducted by executing queries "as is." Once a query optimizer for TP-databases is built (which we are currently working on [7]), the timings reported should improve substantially. With hand-optimized versions of some of the queries, we noticed significant improvements in running time. However, due to space reasons, we have chosen to defer the important topic of query optimization and probabilistic indexes to a future paper [7]. *In some of the charts shown in Appendix A reflecting the results of the experiments, readers may sometimes see only two lines instead of eight, because the four lines denoting the TP-computations and the four lines denoting the TATA-computations are almost identical.*

7.1.1 *Comparing TATA vs. TP-Algebra.* Our experiments were conducted as follows. We generated TP-relations containing $nTuples$ TP-tuples where $nTuples \in \{100, 500, 1000\}$. Each TP-tuple had one TP-case $\langle C_i, D_i, L_i, U_i, \delta_i\rangle$ where $C_i = D_i = (t_1 \sim t_2)$, $t_1 = \mathrm{random}(\{t \in \mathrm{sol}((1/1/1998) \sim (31/12/1998))\})$, t_2 is the time point which occurs $nTimePoints$ days after t_1. Probabilities were assigned randomly. We allowed different probability distributions (independence, geometric, binomial,or a mix of these three) in TP-relations. Using these relations, we calculated the (median of 3) computation times for each of the following operations:

Intersection and Union Computations

$\Xi^{hy}(r \cap_{eq} r')$, $\mathrm{ANN}(r) \cap_{eq} \mathrm{ANN}(r')$, $\Xi^{hy}(r \cup_{eq} r')$, and $\mathrm{ANN}(r) \cup_{eq} \mathrm{ANN}(r')$.

Chart **(a)** in Appendix A shows that intersection takes time that is more or less linear in the number of tuples. Furthermore, as the number of TP-tuples increases, the savings rendered by using TP-tuples instead of annotated tuples increases significantly. Chart **(b)** in Appendix A shows that increasing the total number of time-points (i.e. increasing the effect of uncertainty) has no effect whatsoever on TP-tuples, but the effect on annotated tuples is very significant.

Charts **(a)** and **(b)** jointly show that as far as intersection is concerned, the distributions used have no significant impact on the efficiency of computing intersection. Similar results hold for *union, difference and projection* operations — the reader interested in experimental data for these operations is referred to [9].

Selection Computations

$\sigma_{\mathcal{C}}(r)$ and $\sigma_{\mathcal{C}}(\mathrm{ANN}(r))$ for each type of selection condition $\mathcal{C}$ (i.e., data, temporal, and probabilistic).

We ran three types of experiments with selections involving conditions on data attributes (Charts **(c)** and **(d)**), temporal attributes (Charts **(e)** and **(f)**), and probabilistic attributes (Charts **(g)** and **(h)**), respectively.

When we held the average number of time points per TP-case constant to 16, and increased the number of tuples, we notice that the TP-algebra significantly outperforms the TATA algebra. Furthermore, as the number of data tuples increases, there is very little increase in time on the TP-side, in contrast to the much larger increase on the TATA side. The same phenomenon may be noted when the number of tuples is held constant, but the amount of uncertainty is increased.

An important point to note is that Charts **(g)** and **(h)** indicate that performing probabilistic selections on TP-databases that use uniform distributions is faster than on identical TP-databases that use other distributions!

Join Computations

$\Xi^{hy}(r \bowtie_{\alpha} r')$ and $\mathrm{ANN}(r) \bowtie_{\alpha} \mathrm{ANN}(r')$ for each conjunction strategy $\alpha \in \{ig, pc, nc, in\}$.

We first studied what happens with join under the positive correlation conjunction strategy (Charts **(i)** and **(j)**). Subsequently, we studied what happens with join when we vary the conjunction strategy used. In the first case, we noticed that the performance of TP-join is affected relatively little when we increase number of tuples and/or the the amount of uncertainty. However, as seen in charts Charts **(k)** and **(l)**, using negative correlation as the conjunction strategy is actually much more efficient than using the other strategies, both on the TP and the TATA side — an observation that we have not seen made before. (This is in interesting contrast to previous beliefs that using independence assumptions leads to greater efficiency). For instance, Chart **(k)** shows that when $nTuples = 1 \times 10^3 = 1000$, the computation time for the TATA algebra using $\otimes_{nc}$ ($\otimes_{pc}$) is 157 (2615) seconds while the TP-algebra always finishes under 19 seconds.

7.1.2 *Scalability of TP-Algebra Operations.* We studied the performance of selection, and join as we increased the amount of uncertainty in the data. Charts **(m)** and **(n)** show what happens when we use a mix of distribution functions, and either 100 or 1000 TP-tuples per TP-relation, and vary the number of solutions to TP-cases over the set 4, 96, 5760 and $345,600$. *Due to the size of these numbers, the charts shown use a log-scale.* Chart **(m)** shows the results of performing both selects and joins when we are looking at the case of 100 TP-tuples.

As the reader can see, temporal selections are almost completely unaffected by the amount of uncertainty both in the case of 100 TP-tuples and 1000 TP-tuples (where the time taken stays constant). However, probabilistic selects are expensive to compute (almost as expensive as joins), because they require that the distribution function be applied to all time points in a TP-case. Notice that even when we have 345,600 time-points inside each of these 100 tuples (making up a "flat relation" of size 34,560,000), it takes only about 60 seconds to evaluate the probabilistic select. When we have 345,600 time-points inside each of the 1000 TP-tuples shown in Chart **(n)** (making a flat relation of size 3.5 billion approximately), we see that the time taken is about 125 seconds, reflecting a doubling in the time, though the data increased in size by a factor of 10. We feel this is quite efficient.

Our framework is also quite efficient for computing TP-joins. As can be seen from Chart **(m)**, when we compute a join of two relations consisting of 100 TP-tuples each and 345,600 time-points inside each of these 100 TP-tuples, the join takes about 75 seconds — a bit more expensive than a probabilistic select, but not too bad. When we use a 1000 TP-tuples (and the same 345,600 time-points inside each of these 1000 TP-tuples), the join takes about 580 seconds — a five fold increase when the data tuples in the two joined relations were both increased ten fold.

8. RELATED WORK

8.1 Comparison with Dyreson and Snodgrass

Dyreson and Snodgrass [11] were one of the first to model temporal uncertainty using probabilities by proposing the concept of an *indeterminate instant.* Intuitively, an indeterminate instant is an interval of time-points with an associated probability distribution. They propose an extension of SQL that supports (i) specifying which temporal attributes are indeterminate, (ii) correlation credibility which allows a query to use uncertainty to modify temporal data — for example, by using an EXPECTED value correlation credibility, the query will return a *determinate* relation that retains the most probable time point for the event, (iii) ordering plausibility which is an integer between 1 and 100 where 1 denotes that any possible answer to the query is desired while 100 denotes that only a definite answer is desired, and (iv) specifying that certain temporal intervals are indeterminate. Dyreson and Snodgrass [11] develop a semantics for their version of SQL. In addition, they show how to compute probabilities of temporal relationships such as "event e_1 occurs before event e_2," 'event e_1 occurs at the same time as event e_2," etc., and provide efficient data structures to represent probability mass functions.

Our framework may be viewed as in improvement over the the Dyreson-Snodgrass framework in the following ways. (i) First, [11] presents a version of SQL for temporally indeterminate databases , while we present an *algebra* and prove that all our algebraic operations are correct. Both are clearly needed for a database that supports probabilities over temporal attributes. (ii) The base relations in [11] may be viewed as special cases of TP-relations where the C and D fields are *atomic* *time-interval constraints*. In contrast, our framework allows C and D to be arbitrary (atomic and non-atomic) temporal constraints and so TP-relations can be much more succinct than the base relations used in [11]. (iii) In [11], no explicit lower/upper bounds are considered; all probabilities used are point probabilities. This is a special case of our framework, as point probabilities can be represented by intervals with matching lower and upper bounds. Recall that in 1854 [4], Boole noticed that we must use probability intervals whenever we are ignorant of the relationship between events. (iv) In [11], all PDFs are assumed to be complete, while we allow both complete and incomplete PDFs. In fact, we noticed for the first time that determinate PDFs (all complete PDFs are determinate) guarantee linear time consistency checks for TP-databases. (v) In [11], all indeterminate events are assumed to be independent. This assumption is valid for many applications, and invalid for others. We allow *users* to specify in their query what the relationship between events is. Thus, independence can be used in our framework when appropriate, and other dependencies can be used when deemed appropriate. (vi) Our framework supports a host of operations compaction methods, combination functions, compression functions, and tightening) not considered elswhere.

Conversely, there are some things that can be expressed in the Dyreson-Snodgrass framework [11] which we do not handle — for example, in the current paper, we have assumed tuples have only one indeterminate temporal attribute while [11] allows more than one. Extending the framework to accommodate this is no problem (in fact, we are building an application to track land deeds in Vienna, Austria where there is considerable temporal uncertainty involving many fields) but does make the presentation of the framework more complex. Furthermore, we have no analog of correlation credibility or ordering plausibility. Our experiments complement those of [11] in that we examine how different distributions fundamentally affect the efficiency of the algebraic operations. Note that distribution functions can be stored according to the methods described in [11].

8.2 Relationship with work in Probabilistic Databases

Though there is extensive work on probabilistic databases, there is very little work that merges probabilistic reasoning with time. Kiessling *et al.*'s DUCK system [19; 39; 31] provides an elegant, logical, axiomatic theory for rule based uncertainty. Ng and Subrahmanian [27; 29] have provided a probabilistic semantics for deductive databases — they assume absolute ignorance, and furthermore, assume that rules are present in the system. Lakshmanan and Sadri [25] show how selected probabilistic strategies can be used to extend the previous probabilistic models. Lakshmanan and Shiri [26; 32] have shown how deductive databases may be parametrized through the use of conjunction and disjunction strategies, an approach also followed by Dekhtyar and Subrahmanian [8].

Barbara et al. [3] develop a point probabilistic data model and propose probabilistic operators. In contrast, we allow interval probabilities which permit margins of error in the probability data. In addition, when performing joins, they assume that Bayes' rule applies (and hence, as they admit up front, they make the assumption that all events are independent). Also, as they point out, unfortunately their definition leads to a "lossy" join. Cavallo and Pittarelli [6]'s probabilistic relational database model uses probabilistic projection and join operations, but the other relational algebra operations are not specified.

An important paper on the topic is by Dey and Sarkar [10] who propose an elegant `1NF` approach to handling probabilistic databases. They support (i) having uncertainty about some objects but certain information about others, (ii) first normal form which is easy to understand and use, (iii) elegant new operations like conditionalization, The `1NF` representation used by them is a special case of the annotated representation in this paper — as pointed out by Dyreson and Snodgrass [11], this representation is *not* suitable for directly representing temporal indeterminacy. Many of our operators generalize theirs — for instance, their notion of *union* clusters together all data-identical tuples and takes their `max`, difference clusters together all data-identical tuples and subtracts probability values, and their notion of projection clusters together all data-identical tuples and takes the sum of the tuples' probabilities (or 1, whichever is smaller) to be the probability. These computations are probabilistically legitimate only under some assumptions on the dependencies between the events involved. Our notion of *combination functions* generalize these substantially. In addition, their notion of join only applies under an independence assumption, which we do not require. Similarly, our notion of compaction operations may be viewed as extensions of the two *coalesce* operations proposed by them — we propose whole families of coalesce operations in contrast, and our algebra uses such operations as parameters. Dey and Sarkar [10] propose some operations such as *conditionalization* and N^{th}-Moment that have no analogs in our paper, and deserve further study.

This paper builds on top of the ProbView system for probabilistic databases[23]. ProbView extends the classical relational algebra by allowing users to specify in their query, what probabilistic strategy (or strategies) should be used to parameterize the query. ProbView removed the independence assumption from previous works. However, ProbView has no notion of time, and it was noted by Snodgrass [35] that though ProbView scaled up well to massive numbers of tuples, it did not scale up well when massive amounts of uncertainty are present as is the case with temporal probabilistic databases, where saying that an event sometime between Jan 1-4 yields a total of $4 \times 24 \times 60 \times 60 = 345,600$ seconds. Thus, if our temporal database uses seconds as it lowest level of temporal granularity, this gives rise to $345,600$ cases to represent just one statement — something that would quickly overwhelm ProbView. As the reader can see and as our experiments indicate, TP-databases were specifically designed to eliminate this problem.

8.3 Relationship with work in Temporal Databases

Snodgrass was one of the first to model indeterminate instances in his doctoral dissertation [34] — he proposed the use of a model based on three valued logic. Dutta [15] and Dubois and Prade [13] later used a fuzzy logic based approach to handle *generalized temporal events* — events that may occur multiple times.

Gadia [17] proposes an elegant model to handle incomplete temporal information as well. He models values that are completely known, values that are unknown but are known to have occurred, values that are known if they have occurred, and values that are unknown even if they occurred. However, he makes no use of probabilistic information.

Koubarakis [21] proposes the use of constraints for representing event occurrences. His framework allows stating the facts that event e_1 occurred between 8 and 11 AM, and that event e_2 occurs after 12pm. From this, we may conclude that event e_2 occurs after e_1 — our framework can support this conclusion as well. However, inside our TP-tuples, we cannot state that event e_2 occurs after e_1 — something we can do in a query, but which Koubarakis [21] can explicitly encode in his tuples.

Another important body of work is that of Brusoni *et al.*[5] who developed a system called LaTeR. LaTeR restricts constraints to conjunctions of linear inequalities, as does Koubarakis' work. LaTeR makes a compromise — when tuples are inserted, it builds a constraint network (which increases insertion time), but this pays off because at query time, queries can be efficiently processed. We can benefit from this strategy in our work — as constraint networks are main memory data structures, an adaptation to disk-based structures would greatly enhance scalability. We will report on such efforts in part II of this series of papers [7].

9. CONCLUSIONS

There are a large variety of applications where there is uncertainty about when certain real-world events occurred. Such applications range from shipping and transportation applications, where extensive statistical data is available about shipping times for packages from one location to another, to data mining and time series applications where predictions about when certain stock market activity may occur is inherently uncertain. A variety of other important applications involving uncertainty about when events occur have been identified by Dyreson and Snodgrass in [11].

The only previous work whose explicit goal was to incorporate uncertainty into temporal databases is due to Dyreson and Snodgrass [11]. In this paper, we choose a philosophically different approach to incorporating probabilistic temporal reasoning in relational databases — instead of adding probabilities to temporal databases, we instead add time to probabilistic databases. Our approach allows us to make the following important contributions over and above the important work of Dyreson and Snodgrass [11].

- We propose what is, to our knowledge, the first extension of the relational algebra that integrates both probabilities and time. This nicely complements the probabilistic temporal SQL language designed by Dyreson and Snodgrass [11].
- Second, our framework removes several assumptions made in previous work. First, our framework allows users to specify in their (algebraic) queries, what dependencies (if any) they assume between indeterminate instances. No conditional independence assumptions are required *unless desired* by the user. Instead, the user can parameterize his query with a variety of other probabilistic assumptions. Second, we allow the database to associate *partial distributions* with uncertain data. This is certainly very practical. Most statistical sampling methods do not provide total distributions, but distributions with associated margins of error. Third, by introducing the TP-Algebra, we show how the PDM (Probabilistic Data Model) model can be modified to support temporal indeterminacy, even if there might be several million elements in a set of possible chronons. This was an important open problem raised by Snodgrass in [41].
- We propose two algebras in this paper. The TATA-Algebra is intended for *purely*

theoretical purposes. As the TATA-Algebra explicitly specifies the probability of an event occurring at any given time point, it leads to unacceptably large relations. However, the explicit specification allows us to easily specify *how* the relational operations should be defined, i.e. what their behavior should be so as to be "probabilistically and temporally kosher."

The TP-Algebra on the other hand is an *implementation oriented algebra.* First, TP-relations are very small compared to annotated relations. Second, for every operation *op* defined on the TATA-Algebra, we show how to define an analogous operation that directly manipulates the succinct TP-relations. We show that these TP-operations are all *correct* in the sense that they correctly implement the TATA-Algebra operations. **Thus, there is no need to implement the TATA-Algebra because the TP-Algebra can realize it in a sound, complete, and much more efficient manner.**

• We provide a host of new algebraic operations that have not been introduced before. These include a variety of *compaction operators*, *compression operators*, *combination operators*, and a *tightening* operator.

• We have conducted experiments on the feasibility of our approach by building a prototype TP-Algebra system on top of ODBC. Our experiments show that the distributions that are used definitely impact the performance of the system. TP-relations are shown to be far more scalable than their annotated counterparts.

ACKNOWLEDGMENTS

The authors would like to thank the annonymous reviewer for a number of very helpful comments.

REFERENCES

[1] J.F. Allen. (1984) *Towards a General Theory of Time and Action*, Artificial Intelligence, 23, pps 123–154.

[2] J.F. Baldwin. (1987) *Evidential Support Logic Programming*, J. of Fuzzy Sets and Systems, 24, pps 1-26.

[3] D. Barbara, H. Garcia-Molina and D. Porter. (1992) *The Management of Probabilistic Data*, IEEE Trans. on Knowledge and Data Engineering, Vol. 4, pps 487–502.

[4] G. Boole. (1854) *The Laws of Thought*, Macmillan, London.

[5] V. Brusoni, L. Console, P. Terenziani and B. Pernici. (1995) *Extending temporal relational databases to deal with imprecise and qualitative temporal information*, Proc. Intl. Workshop on Recent Advances in Temporal Databases (eds. S. Clifford and A. Tuzhilin), pps 3–22, Springer Verlag.

[6] R. Cavallo and M. Pittarelli. (1987) *The Theory of Probabilistic Databases*, Proc. VLDB 1987.

[7] A. Dekhtyar, R. Ross and V.S. Subrahmanian. (1999) *Probabilistic Temporal Databases, II: Indexes, Query Optimization and Updates*, in preparation.

[8] A. Dekhtyar and V.S. Subrahmanian. (1997) *Hybrid Probabilistic Logic Programs*, Proc. 1997 Intl. Conf. on Logic Programming (ed. L. Naish), MIT Press.

[9] A. Dekhtyar, R. Ross, and V.S. Subrahmanian. (1998) *Probabilistic Temporal Databases*, University of Maryland Technical Report, CS-TR-3987. Available at http://www.cs.umd.edu:80/Dienst/UI/2.0/Describe/ncstrl.umcp/CS-TR-3987?abstract=Dekhtyar.

[10] D. Dey and S. Sarkar. (1996) *A Probabilistic Relational Model and Algebra*, ACM Transactions on Database Systems, Vol. 21, 3, pps 339–369.

[11] C. Dyreson and R. Snodgrass. (1998) *Supporting Valid-Time Indeterminacy*, ACM Transactions on Database Systems, Vol. 23, Nr. 1, pps 1—57.

[12] D. Dubois and H. Prade. Certainty and Uncertainty of Vague Knowledge and Generalized Dependencies in Fuzzy Databases. In *Proceedings International Fuzzy Engineering Symposium*, pp. 239–249, Yokohama, Japan, 1988.

[13] D. Dubois and H. Prade. (1989) *Processing Fuzzy Temporal Knowledge*, IEEE Transactions on Systems, Man and Cybernetics, 19, 4, pps 729–744.

[14] R. O. Duda, P. E. Hart and N. J. Nilsson. (1976) *Subjective Bayesian Methods for Rule-based Inference Systems*, Proceedings of National Computer Conference, pp 1075–1082.

[15] S. Dutta. (1989) *Generalized Events in Temporal Databases* , in: Proc. 5th Intl. Conf. on Data Engineering, pps 118–126.

[16] R. Fagin, J. Y. Halpern and N. Megiddo. (1988) *A logic for reasoning about probabilities*, Information and Computation, 87(1/2):78-128, July/August 1990.

[17] S. Gadia, S. Nair and Y.C. Poon. (1992) *Incomplete Information in Relational Temporal Databases*, Proc. Intl. Conf. on Very Large Databases.

[18] U. Guntzer, W. Kiessling and H. Thone. (1991) *New Directions for Uncertainty Reasoning in Deductive Databases*, Proc. 1991 ACM SIGMOD, pp 178–187.

[19] W. Kiessling, H. Thone and U. Guntzer. (1992) *Database Support for Problematic Knowledge*, Proc. EDBT-92, pps 421–436, Springer LNCS Vol. 580.

[20] Henry Korth and Abraham Silberschatz. (1991) *Database System Concepts, Second Edition*, McGraw-Hill Inc.

[21] M. Koubarakis. (1994) *Database Models for Infinite and Indefinite Temporal Information*, Information Systems, Vol. 19, 2, pps 141–173.

[22] S. Kraus, Y. Sagiv and V.S. Subrahmanian. (1996) Representing and Integrating Multiple Calendars. University of Maryland Technical Report CS-TR-3751. Submitted for journal publication.

[23] V.S. Lakshmanan, N. Leone, R. Ross and V.S. Subrahmanian. ProbView: A Flexible Probabilistic Database System. ACM Transactions on Database Systems, Vol. 22, Nr. 3, pps 419–469, Sep. 1997.

[24] V.S. Lakshmanan and F. Sadri. (1994) *Modeling Uncertainty in Deductive Databases*, Proc. Int. Conf. on Database Expert Systems and Applications, (DEXA'94), September 7-9, 1994, Athens, Greece, Lecture Notes in Computer Science, Vol. 856, Springer (1994), pp. 724-733.

[25] V.S. Lakshmanan and F. Sadri. (1994) *Probabilistic Deductive Databases*, Proc. Int. Logic Programming Symp., (ILPS'94), November 1994, Ithaca, NY, MIT Press.

[26] V.S. Lakshmanan and N. Shiri. (1997) *A Parametric Approach with Deductive Databases with Uncertainty*, accepted for publication in IEEE Transactions on Knowledge and Data Engineering.

[27] R. Ng and V.S. Subrahmanian. (1993) Probabilistic Logic Programming, Information and Computation, 101, 2, pps 150–201, 1993.

[28] R. Ng and V.S. Subrahmanian. A Semantical Framework for Supporting Subjective and Conditional Probabilities in Deductive Databases, Journal of Automated Reasoning, 10, 2, pps 191–235, 1993.

[29] R. Ng and V.S. Subrahmanian. (1995) Stable Semantics for Probabilistic Deductive Databases, Information and Computation, 110, 1, pps 42-83.

[30] H. J. Samet. (1989) *The Design and Analysis of Spatial Data Structures*, Addison Wesley.

[31] H. Schmidt, W. Kiessling, U. Guntzer and R. Bayer. (1987) *Combining Deduction by Uncertainty with the Power of Magic*, Proc. DOOD-89, pps 205–224, Kyoto, Japan.

[32] N. Shiri. (1997) *On a Generalized Theory of Deductive Databases*, Ph.D. Dissertation, Concordia University, Montreal, Canada, August 1997.

[33] J. Shoenfield. (1967) *Mathematical Logic*, Addison Wesley.

[34] R.T. Snodgrass. (1982) *Monitoring Distributed Systems: A Relational Approach*, PhD dissertation, Carnegie Mellon University.

[35] R.T. Snodgrass. (1996) Personal communication to V.S. Subrahmanian.

[36] V.S. Subrahmanian. (1987) *On the Semantics of Quantitative Logic Programs*, Proc. 4th IEEE Symp. on Logic Programming, pps 173-182, Computer Society Press. Sep. 1987.

[37] V.S. Subrahmanian. (1988) *Generalized Triangular Norm and Co-Norm Based Semantics for Quantitative Rule Set Logic Programming*, Logic Programming Research Group Technical Report LPRG-TR-88-22, Syracuse University.

[38] V.S. Subrahmanian. (1998) *Principles of Multimedia Database Systems*, Morgan Kaufmann.

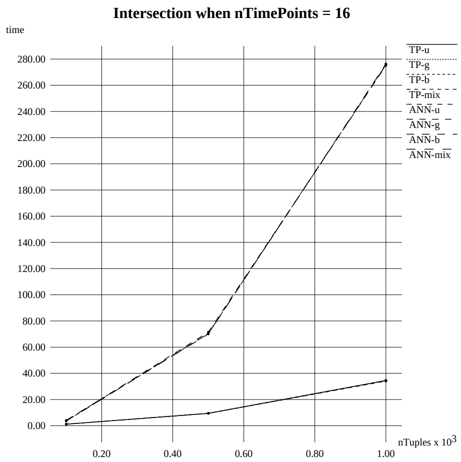
[39] H. Thone, W. Kiessling and U. Guntzer. (1995) *On Cautious Probabilistic Inference and Default Detachment*, Annals of Operations Research, 55, pps 195–224.

[40] J.D. Ullman. (1989) *Principles of Database and Knowledge Base Systems*, Computer Science Press, 1989.

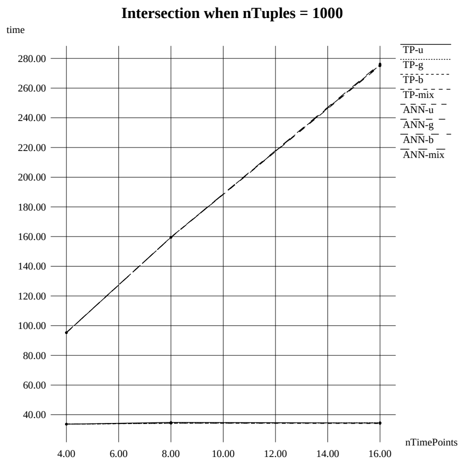
[41] C. Zaniolo, S. Ceri, C. Faloutsos, R. Snodgrass, V.S. Subrahmanian, and C. Zicari. (1997) *Advanced Database Systems*, Morgan Kaufman.

APPENDIX

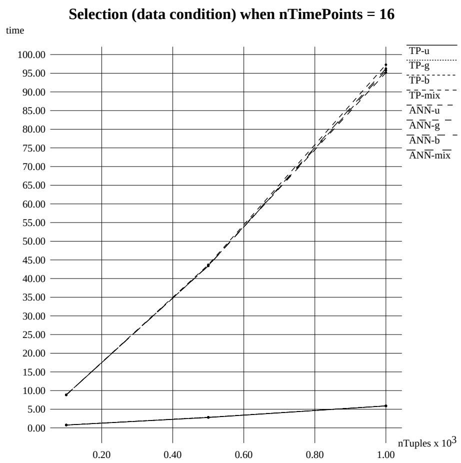
A. APPENDIX: EXPERIMENTAL RESULTS



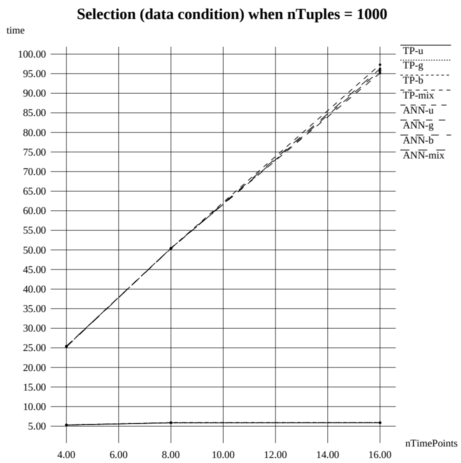
(a)



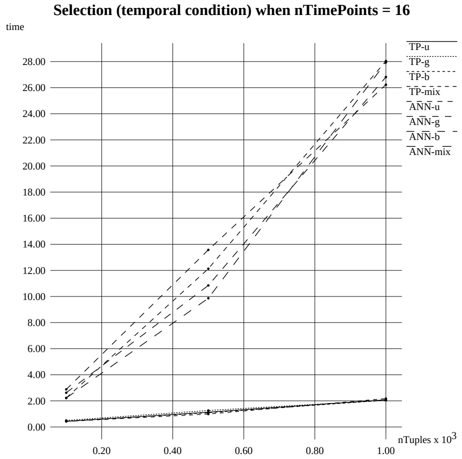
(b)



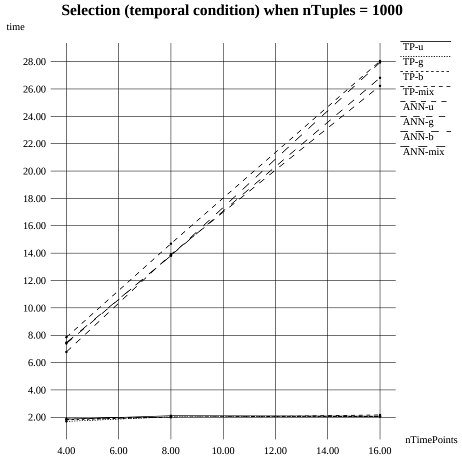
(c)



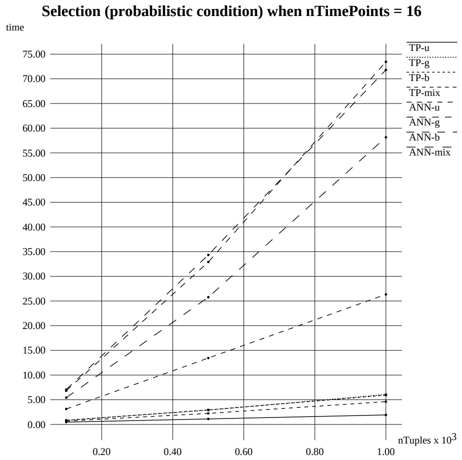
(d)


Selection (temporal condition) when nTimePoints = 16
time
TP-u
TP-g
TP-b
TP-mix
ANN-u
ANN-g
ANN-b
ANN-mix
nTuples x 10^3

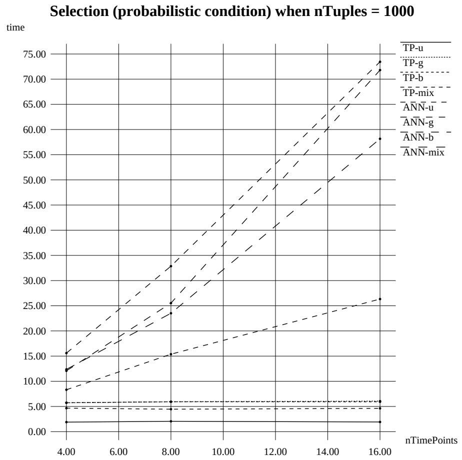
(e)


Selection (temporal condition) when nTuples = 1000
time
TP-u
TP-g
TP-b
TP-mix
ANN-u
ANN-g
ANN-b
ANN-mix
nTimePoints

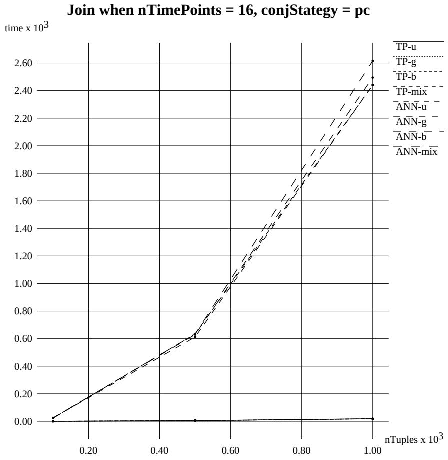
(f)


Selection (probabilistic condition) when nTimePoints = 16
time
TP-u
TP-g
TP-b
TP-mix
ANN-u
ANN-g
ANN-b
ANN-mix
nTuples x 10^3

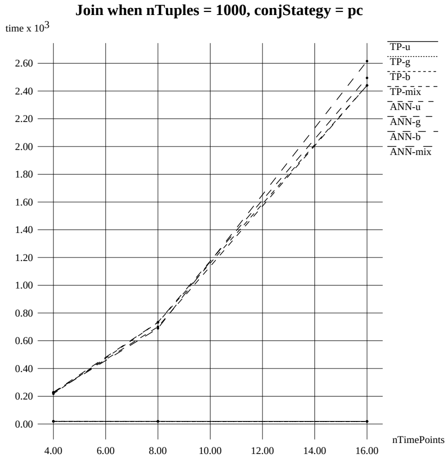
(g)


Selection (probabilistic condition) when nTuples = 1000
time
TP-u
TP-g
TP-b
TP-mix
ANN-u
ANN-g
ANN-b
ANN-mix
nTimePoints

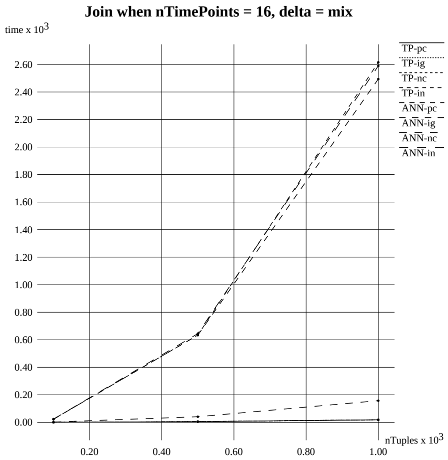
(h)


Join when nTimePoints = 16, conjStategy = pc
time x 10^3
nTuples x 10^3
TP-u
TP-g
TP-b
TP-mix
ANN-u
ANN-g
ANN-b
ANN-mix

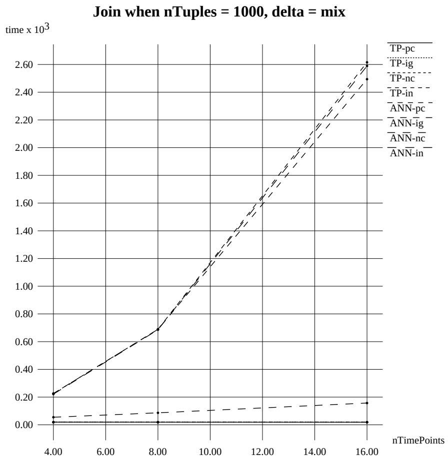
(i)


Join when nTuples = 1000, conjStategy = pc
time x 10^3
nTimePoints
TP-u
TP-g
TP-b
TP-mix
ANN-u
ANN-g
ANN-b
ANN-mix

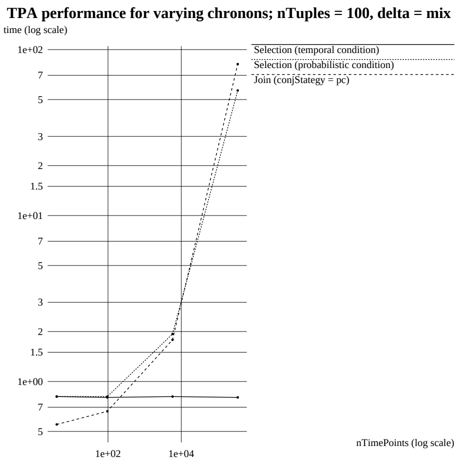
(j)


Join when nTimePoints = 16, delta = mix
time x 10^3
nTuples x 10^3
TP-pc
TP-ig
TP-nc
TP-in
ANN-pc
ANN-ig
ANN-nc
ANN-in

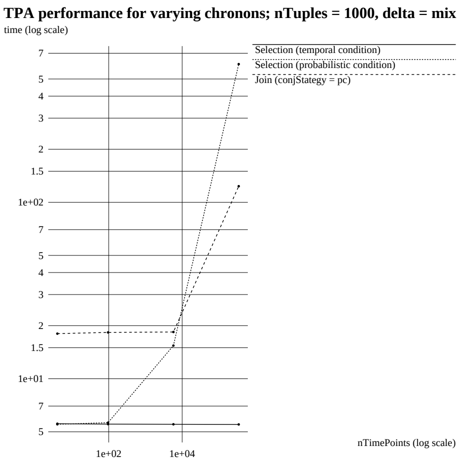
(k)


Join when nTuples = 1000, delta = mix
time x 10^3
nTimePoints
TP-pc
TP-ig
TP-nc
TP-in
ANN-pc
ANN-ig
ANN-nc
ANN-in

(l)


TPA performance for varying chronons; nTuples = 100, delta = mix
time (log scale)
Selection (temporal condition)
Selection (probabilistic condition)
Join (conjStategy = pc)
nTimePoints (log scale)
1e+02
1e+04
1e+00
1e+01

(m)


TPA performance for varying chronons; nTuples = 1000, delta = mix
time (log scale)
Selection (temporal condition)
Selection (probabilistic condition)
Join (conjStategy = pc)
nTimePoints (log scale)
1e+02
1e+04
1e+01

(n)