Probabilistic Temporal Databases, I: Algebra

Alex Dekhtyar\footnote{Dept. of Computer Science, University of Maryland, College Park, MD 20742. Email: dekhtyar@cs.umd.edu.} Robert Ross\footnote{Dept. of Computer Science, University of Maryland, College Park, MD 20742. Email: robross@cs.umd.edu.} V. S. Subrahmanian\footnote{Dept. of Computer Science, Institute for Advanced Computer Studies and Institute for Systems Research, University of Maryland, College Park, MD 20742. Email: vs@cs.umd.edu.}

January 26, 1999

Abstract

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 [53], 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.

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 waypoint, 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.

- **Video Extraction**: A completely different application arises in the case of feature extraction in video databases [49]. A video may be viewed as a sequence of frames (still images). A video feature extraction algorithm attempts to identify objects and activities occurring in these frames. However, most image processing algorithms are uncertain in their identifications. Thus, the statement “Darth Vader appears in frames 26 and 27” is an uncertain statement about uncertainty and time (frame numbers are correlated tightly with time as most video players in the market today playback video at a rate of 15 to 30 frames per second). Thus, the creation of a video database which automatically identifies features and/or gestures encompasses some temporal aspects as well as some uncertainty.

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 must 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 must 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 case of the stock market application, we must 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.

- **We first 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.

- **We then 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) 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 then define 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 \( \alpha \) in the Theoretical Annotated Temporal Algebra, there is a corresponding operation \( \alpha' \) 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.

- We show that each operator, whether in the TP-Algebra, or in the Theoretical Annotated Temporal Algebra, can be parametrized by the user’s knowledge of the dependencies between events. This is important because, as shown in [30], the probability of a complex event like \((e_1 \lor e_2)\) depends upon our knowledge of the dependencies between \(e_1\) and \(e_2\).

- We present an implementation of the TP-Algebra on top of ODBC and provide a set of experimental results.

The idea of handling uncertainty in temporal databases was first addressed by Dyreson and Snodgrass [13]. They proposed the concept of an indeterminate instant where we know that an event occurs at some point in a set of time points, but we do not know exactly when. However, a probability distribution is known. Dyreson and Snodgrass [13] propose an extension of SQL to handle indeterminate valid-time, and show that their implementation is “reliable” (correct). They provide elegant data structures to represent probability mass functions, and algorithms to compute temporal relationships between indeterminate events. The authors provide impressive experimental results. The model presented in [13] proceeds under the following assumptions, all of which are removed in our framework.

- “All indeterminate instances are considered to be independent” [13, p.7]. In our paper, we show how the user can explicitly specify in his query, his knowledge of the dependency or lack thereof between events. Thus, this assumption is eliminated by us.

- “… we do not allow partially known distributions” [13, p.8]. In this paper, we will allow partial distributions to be specified.

- “We could not adopt the PDM approach or its successors to support temporal indeterminacy, since there might be several million elements in a set of possible chronons. Representing each alternative with an associated probability is impractical.” [13, p.46]. Our TP-Algebra explicitly shows how to get around this problem.

Last but not least, our paper provides an extension of the relational algebra to handle temporal probabilistic data — the contributions of [13] present an extension of SQL, thus neatly complementing our work. As we will see in Section 8, there are many other interesting and important contributions made in [13] that complement the work reported here, leading to the potential for a very powerful system obtained by combining the two frameworks.

The relationship between relational, temporal, and TP-databases may be briefly summed up as follows. In classical relational databases [51] a data-relation \( R \) over schema \( (A_1, \ldots, A_n) \) contains a set of tuples, as shown in Figure 1(a). In contrast, a temporal database relation \( R \) may be thought of as shorthand for a set
of snapshots $R(1), R(2), \ldots$ — intuitively, given a time instant $i$, $R(i)$ is a data-relation that specifies what tuples are true w.r.t. relation $R$ at time $i$. This is shown in Figure 1(b).

In contrast, a TP-relation is much more complicated, as shown in Figure 1(c). For every given point $i$ in time, we cannot specify $R(i)$ precisely as we are uncertain about what is in $R$ at time $i$. Thus, for each time instant $i$, a TP-relation specifies a set of relations, $\{R(i, 1), \ldots, R(i, k_i)\}$ for some $k_i > 0$, together with a probability assignment, $\varphi$ on $\{R(i, 1), \ldots, R(i, k_i)\}$. Intuitively, $\varphi(R(i, j)) = 0.3$ means that there is a 30% probability that the content of relation $R$ at time $i$ is $R(i, j)$.

## 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 Subsections 2.1, 2.2, 2.3 and refsec:ProbStrat is not new work, but form the basic definitions needed to describe our algebras. Other parts of this section describe new work.

- We first define (Sec. 2.1) a calendar, borrowing from definitions in [29]. Calendars are needed because all TP-relations will assume that time is specified with respect to an arbitrary but fixed calendar.

- Then, we define (Sec. 2.2) what a temporal constraint over an arbitrary but fixed calendar is. As specified earlier in the paper, our algebras use constraints to describe sets of time points.

- We then define (Sec. 2.3) what distribution functions are.

- Next, we define (Sec. 2.4) what a probabilistic tuple is. This definition will serve as a “springboard” for the later definitions.
• In Section 2.5, we specify a set of axioms that a function must satisfy for it to be considered a probabilistic conjunction or disjunction strategy. For example, when we compute the cartesian product of two TP-relations $R$ and $S$ where TP-tuples $r \in R$ and $s \in S$ at time $i$ with probabilities in the interval $[pr_1, pr_2]$ and $[ps_1, ps_2]$ respectively, then a probabilistic conjunction strategy allows us to compute the probability that the concatenation $r \odot s$ is in the cartesian product of $R, S$. Clearly, this probability depends upon our knowledge (if any) of the dependencies between the events denoted by these tuples. Section 2.5 specifies axioms that a function must satisfy for it to be considered a probabilistic conjunction or disjunction strategy. When a user of a TP-database asks a query, he may ask the system to execute the operations in his query under a probabilistic strategy (or strategies) that he believes captures the relationship between the events involved.

• Finally, in Section 2.6, we specify the means by which 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 [29].

**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 $\{\text{all integers}\}$ 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 $\subseteq$ among those time units. The greatest time unit according to $\subseteq$ may be either finite or infinite, while all other time units in the hierarchy must be finite.

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

**Definition 2.3 (time point)** Suppose $T_1 \subseteq \cdots \subseteq T_n$ 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 $(v_1/\ldots/v_n)$ 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 $t = (v_{\text{day}}, v_{\text{month}}, v_{\text{year}}) \leq_H t' = (v'_{\text{day}}, v'_{\text{month}}, v'_{\text{year}})$, is true iff 

$((v_{\text{year}} < v'_{\text{year}}) \lor (v_{\text{year}} = v'_{\text{year}} \land v_{\text{month}} < v'_{\text{month}}) \lor (v_{\text{year}} = v'_{\text{year}} \land v_{\text{month}} = v'_{\text{month}} \land v_{\text{day}} \leq v'_{\text{day}}))$.

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

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

Let $\text{next}_\tau(t)$ denote the next, consecutive time point after $t$. Thus, $\text{next}_\tau(t)$ 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 $\delta$”, 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 [29]) how temporal constraints can be used to specify sets of time points associated with a calendar.

**Definition 2.5 (atomic temporal constraint)** Suppose $T_1 \subseteq \cdots \subseteq T_n$ is a linear hierarchy $H$ of time units over calendar $\tau$. An atomic temporal constraint over calendar $\tau$ 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 $(t_i)$ be an abbreviation for $(t_i \sim t_i)$.

For example, $(\text{day} < 15), (\text{month} \geq 8)$, and $(12/3/1997 \sim 10/4/1997)$ are all atomic temporal constraints, but $(1996 = \text{year})$ is not. Also, $(\text{day} < 45)$ is not an atomic temporal constraint since 45 is not in the time-value set of day. Similarly, $(15/2/1997 \sim 29/2/1997)$ is not an atomic temporal constraint since $(29/2/1997)$ is not a valid time point in $\tau$. Furthermore, $(10/4/1997 \sim 12/3/1997)$ 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 $\tau$ is defined inductively in the following way:

- Any atomic temporal constraint over $\tau$ is a temporal constraint over $\tau$. 


If \( C_1 \) and \( C_2 \) are temporal constraints over \( \tau \), then \((C_1 \land C_2)\), \((C_1 \lor C_2)\), and \((\neg C_1)\) are temporal constraints over \( \tau \).

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, \(((day > 5 \land day < 15) \land (month = 4 \lor month \geq 8) \land year = 1996)\) and \(((12/3/1997 \sim 10/4/1997) \lor (10/7/1997 \sim 10/7/1997))\) are temporal constraints but \((day > 5 \land day < 15)\) is not.

**Definition 2.7 (solution set to an atomic temporal constraint)** Suppose \( T_1 \subseteq \cdots \subseteq T_n \) is a linear hierarchy \( H \) of time units over calendar \( \tau \). Then an atomic temporal constraint over \( \tau \) 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 \( \tau \) is the set \( S \) which is defined in the following way:

<table>
<thead>
<tr>
<th>Case</th>
<th>( S )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( op = (\leq) )</td>
<td>( { t \mid t \in S_\tau \land t.T_i \leq T_i \ v_i } )</td>
</tr>
<tr>
<td>( op = (&lt;) )</td>
<td>( { t \mid t \in S_\tau \land t.T_i &lt; T_i \ v_i } )</td>
</tr>
<tr>
<td>( op = (=) )</td>
<td>( { t \mid t \in S_\tau \land t.T_i = T_i \ v_i } )</td>
</tr>
<tr>
<td>( op = (#) )</td>
<td>( { t \mid t \in S_\tau \land t.T_i \neq T_i \ v_i } )</td>
</tr>
<tr>
<td>( op = (&gt;) )</td>
<td>( { t \mid t \in S_\tau \land t.T_i &gt; T_i \ v_i } )</td>
</tr>
<tr>
<td>( op = (\geq) )</td>
<td>( { t \mid t \in S_\tau \land T_i \leq T_i \ v_i } )</td>
</tr>
<tr>
<td>( op = (~) )</td>
<td>( { t \mid t \in S_\tau \land t.T_i \leq_H t } )</td>
</tr>
</tbody>
</table>

For instance, the solution set to \((day > 25)\) over the Gregorian calendar \( \tau \) 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 \( S_\tau \). 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.

**Definition 2.8 (solution set to a temporal constraint)** Let \( S_\tau \) be the set of all valid time points over calendar \( \tau \). Then the solution set to a temporal constraint \( C \) over calendar \( \tau \), abbreviated \( \text{sol}(C) \), is the set \( S \) which is defined inductively in the following way:

- If \( C \) is an atomic temporal constraint, then \( S = \text{sol}(C) \).
- If \( C \) is of the form \((C_1 \land C_2)\), then \( S = \text{sol}(C_1) \cap \text{sol}(C_2) \).
- If \( C \) is of the form \((C_1 \lor C_2)\), then \( S = \text{sol}(C_1) \cup \text{sol}(C_2) \).
- If \( C \) is of the form \((\neg C_1)\), then \( S = S_\tau - \text{sol}(C_1) \).

Each time point \( t \in \text{sol}(C) \) is called a solution to \( C \).

For example, the solution set to \(((5/8/1997 \sim 10/8/1997) \lor (7/8/1997 \sim 12/8/1997))\) 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.

Definition 2.9 (finite calendar) Calendar $\tau$ is a finite calendar iff $S_\tau$ is finite. \hfill $\Box$

Note that when the greatest time unit of a calendar is finite, then the calendar is guaranteed to be finite. Furthermore, for all temporal constraints $C$ over a finite calendar $\tau$, $\text{sol}(C)$ must be a finite subset of $S_\tau$. Given a finite calendar $\tau$, we use $t_S^{\tau}$ to denote the smallest time point of $\tau$ (w.r.t. the ordering $<_H$ associated with the calendar) and $t_E^{\tau}$ to denote the largest time point. When $\tau$ is clear from context, we will drop the superscript $\tau$ 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 2.10 (probability distribution function) Let $D$ be a temporal constraint over calendar $\tau$ such that $|\text{sol}(D)| \geq 1$. Then a probability distribution function (PDF) over calendar $\tau$, denoted $\text{pdf}(D, t_j)$, is a function which takes $D$ and a time point $t_j \in S_\tau$ as input, and returns as output a probability $\rho_j$ 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 $t_j \notin \text{sol}(D)$, $\rho_j = 0$.
3. $\sum_{t_j \in S_\tau} (\rho_j) \leq 1$. This implies that $\sum_{t_j \in \text{sol}(D)} (\text{pdf}(D, t_j)) \leq 1$.

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. \hfill $\Box$

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 $t_j \in \text{sol}(D)$. Partial PDFs are useful when modeling infinite distributions; here, we are considering only a finite portion of the total probability mass. Determine 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 $S_\tau$ 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 $\tau$, denoted $\text{pdf}_\text{u}(D, t_j)$, is defined as $\rho_j = \frac{1}{|\text{sol}(D)|}$ if $t_j \in \text{sol}(D)$ or $\rho_j = 0$ otherwise. $\text{pdf}_\text{u}$ is a complete PDF.

Notice that for all $t_1, t_2 \in \text{sol}(D)$, $\rho_1 = \rho_2$. 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 $n = |\text{sol}(D)|$
non-zero \( \rho_i \)'s, one for each \( t_j \in \text{sol}(D) \), and \( n \cdot \rho_j = |\text{sol}(D)| \cdot \frac{1}{\text{sol}(D)} = 1 \). Furthermore, we will never divide by zero since by definition of PDFs, \(|\text{sol}(D)| \geq 1\).

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 \( S_\tau \) where \( \text{sol}(D) = \{t_0, \ldots, t_n\} \) and \( t_i \) occurs before \( t_{i+1} \) for all \( 0 \leq i < n \), i.e. \( \text{sol}(D) \) 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 \( \tau \), denoted \( \text{pdf}_{g,p}(D, t_j) \), is defined as \( \rho_j = p \cdot (1 - p)^{t_j} \) if \( t_j = t_i \in \text{sol}(D) \) or \( \rho_j = 0 \) otherwise. \( \text{pdf}_{g,p} \) is a partial PDF. Note that if \(|\text{sol}(D)| \) is fixed (or constant time computable), then \( \text{pdf}_{g,p} \) is a determinate PDF.

If \( p = \frac{1}{3} \), \( \text{pdf}_{g,p}(D, t_0) = \frac{1}{3} \cdot \left(\frac{2}{3}\right)^0 \), \( \text{pdf}_{g,p}(D, t_1) = \frac{1}{3} \cdot \left(\frac{2}{3}\right)^1 \), and \( \text{pdf}_{g,p}(D, t_2) = \frac{1}{3} \cdot \left(\frac{2}{3}\right)^2 \). For all other time points \( t_j \in S_\tau \), \( \text{pdf}_{g,p}(D, t_j) = 0 \). Notice that if \( p = \frac{1}{2} \), then \( \text{pdf}_{g,p}(D, t_0) = \frac{1}{2} \) and \( \text{pdf}_{g,p}(D, t_i) \) will be half of \( \text{pdf}_{g,p}(D, t_{i-1}) \) for each \( 1 \leq i \leq n \).

Let \( \text{pdf}_{g,p} \) be defined in the same way as \( \text{pdf}_{g,p} \) except \( \text{pdf}_{g,p}(D, t_n) = 1 \) if \(|\text{sol}(D)| = 1 \) or \( \text{pdf}_{g,p}(D, t_{i-1}) = 1 - (\sum_{t_j \in \{t_0, \ldots, t_{i-1}\}} (\text{pdf}_{g,p}(D, t_j))) \) otherwise. We call \( \text{pdf}_{g,p} \) the complete correlate of \( \text{pdf}_{g,p} \) since \( \text{pdf}_{g,p}(D, t_j) = \text{pdf}_{g,p}(D, t_j) \) for all \( t_j \in S_\tau - \{t_n\} \) and since \( \text{pdf}_{g,p} \) 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 \(|\text{sol}(D)| > 1 \), \( \text{pdf}_{g,p} \) has the nice property that \( \text{pdf}_{g,p}(D, t_{n-1}) = \text{pdf}_{g,p}(D, t_{n-1}) \).

**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 \( \tau \), denoted \( \text{pdf}_{b,p}(D, t_j) \), is defined as \( \rho_j = \binom{n}{t_j} \cdot p^{t_j} \cdot (1 - p)^{n-t_j} \) if \( t_j = t_i \in \text{sol}(D) \) or \( \rho_j = 0 \) otherwise. \( \text{pdf}_{b,p} \) is a determinate PDF.

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

Techniques that specify how to associate and store probability distributions with events are provided by Dyreson and Snodgrass [13, p. 8] and by Dey and Sarkar [12]. Hence, we do not discuss this matter in further detail here.

Throughout the rest of this paper, we will use \( (\delta = \text{"a"}) \), \( (\delta = \text{"g", p"}) \), \( (\delta = \text{"g", p"}) \), \( (\delta = \text{"p", \lambda"}) \) to represent the distribution functions for \( \text{pdf}_{a} \), \( \text{pdf}_{g,p} \), \( \text{pdf}_{g,p} \), \( \text{pdf}_{b,p} \), and \( \text{pdf}_{p,\lambda, \lambda} \) respectively. Furthermore, unless we specifically state otherwise, assume that parameter \( p = 0.5 \). Thus, \( (\delta = \text{"g"}) \) represents the \( \text{pdf}_{g,0.5} \) function.

### 2.4 P-tuples

In this section, we will briefly introduce the concept of a probabilistic tuple — one that extends the notion of an ordinary tuple to include probabilistic information.

**Definition 2.11 (P-tuple)** Let \( D \) be a temporal constraint over \( \tau \) where \(|\text{sol}(D)| \geq 1 \). Furthermore, let \( L, U \in [0, 1] \) be probabilities where \( L \leq U \), and let \( \delta \) be a distribution function over \( \tau \). Then the quadruple \( \langle D, L, U, \delta \rangle \) is called a probabilistic tuple or P-tuple.
A P-tuple $pt$ is usually associated with an event $e$. Here, $pt$ has the following interpretation: “The probability that $e$ occurred during the time periods described by sol($D$) lies within the interval $[L_t, U_t]$ and is distributed according to $\delta$.” Thus for each $t \in$ sol($D$), $pt$ indicates that event $e$ occurred at time $t$ with probability $P_t \in [L_t, U_t]$.

For instance, let $e$ be the event “packet.47 arrives in Rome”, let $D = (1/8/1997 \sim 3/8/1997)$, and let $pt = (D, 0.4, 0.8, g)$. Here, $pt$ indicates that we should distribute $[0.4, 0.8]$ among the members of sol($D$) according to the geometric PDF. Thus, packet.47 arrived at time $t = (1/8/1997)$ with $[L_t, U_t] = [0.2, 0.4]$, at $t = (2/8/1997)$ with $[L_t, U_t] = [0.1, 0.2]$, or at $t = (3/8/1997)$ with $[L_t, U_t] = [0.05, 0.1]$.

An event is instantaneous if it can only occur at a single point in time. For example, consider the event “Toss $id$ of coin $C$ comes up heads.” This is an instantaneous event since 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. 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 $s(t(e))$ denoting the start of $e$ and the event $e(\cap)e$ denoting the end of $e$. Thus in this paper, without loss of generality, we only consider events that are instantaneous. A similar assumption is made by Dyreson and Snodgrass[13].

### 2.5 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 $(e_1 \land e_2)$? As argued in [30], the answer depends 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 $(e_1 \lor e_2)$. We address these problems by consulting probabilistic conjunction strategies and probabilistic disjunction strategies. Both of these concepts were originally defined in [30] 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 ($L_1 \leq L_2 \land U_1 \leq U_2$).
2. $[L_1, U_1] \geq [L_2, U_2]$ iff ($L_1 \geq L_2 \land U_1 \geq U_2$).
3. $[L_1, U_1] \leq [L_2, U_2]$ iff ($L_1 \geq L_2 \land U_1 \leq U_2$).
4. $[L, U] = ([L_1, U_1] \cap [L_2, U_2])$ iff ($L = \max(L_1, L_2) \land U = \min(U_1, U_2) \land L \leq U$).

**Definition 2.12 (probabilistic conjunction strategy)** Let events $e_1$, $e_2$ have probabilistic intervals $[L_1, U_1]$ and $[L_2, U_2]$ respectively. Then a probabilistic conjunction strategy is a binary operation $\otimes$ which uses this information to compute the probabilistic interval $[L, U]$ for event $(e_1 \land e_2)$. When the events involved are clear from context, we use “$[L, U] = [L_1, U_1] \otimes [L_2, U_2]$” to denote “$(e_1 \land e_2, [L, U]) = (e_1, [L_1, U_1]) \otimes (e_2, [L_2, U_2])$”. Every conjunctive strategy must conform to the following probabilistic postulates:

1. **Bottomline:** $([L_1, U_1] \otimes [L_2, U_2]) \leq [\min(L_1, L_2), \min(U_1, U_2)]$.
2. **Ignorance:** $([L_1, U_1] \otimes [L_2, U_2]) \subseteq [\max(0, L_1 + L_2 - 1), \min(U_1, U_2)]$. A brief explanation of this axiom is in order. Boole proved in 1854 [7] 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 for $(e_1 \land e_2)$ is that it lies in the interval shown above. This forms the basis for numerous pieces of work in the AI and deductive database
literature ([19, 34, 36] 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.

3. **Identity**: When \((e_1 \land e_2)\) is consistent and \([L_2, U_2] = [1, 1], ([L_1, U_1] \otimes [L_2, U_2]) = [L_1, U_1]\).

4. **Annihilator**: \((([L_1, U_1] \otimes [0, 0]) = [0, 0]\).

5. **Commutativity**: \((([L_1, U_1] \otimes [L_2, U_2]) = ([L_2, U_2] \otimes [L_1, U_1])\).

6. **Associativity**: \(((([L_1, U_1] \otimes [L_2, U_2]) \otimes [L_3, U_3]) = ([L_1, U_1] \otimes ([L_2, U_2] \otimes [L_3, U_3]))\).

7. **Monotonicity**: \(([L_1, U_1] \otimes [L_2, U_2]) \leq ([L_1, U_1] \otimes [L_3, U_3])\) if \([L_2, U_2] \leq [L_3, U_3]\).

The following are some sample conjunctive strategies:

- **Use the \(\otimes_{ig}\)** (ignorance) operator when we do not know the dependencies between \(e_1\) and \(e_2\).
  \(([L_1, U_1] \otimes_{ig} [L_2, U_2]) = [\max(0, L_1 + L_2 - 1), \min(U_1, U_2)]\).

- **Use the \(\otimes_{pc}\)** (positive correlation) operator when the overlap between \(e_1\) and \(e_2\) is maximal.
  \(([L_1, U_1] \otimes_{pc} [L_2, U_2]) = [\min(L_1, L_2), \min(U_1, U_2)]\).

- **Use the \(\otimes_{nc}\)** (negative correlation) operator when the overlap between \(e_1\) and \(e_2\) is minimal.
  \(([L_1, U_1] \otimes_{nc} [L_2, U_2]) = [\max(0, L_1 + L_2 - 1), \max(0, U_1 + U_2 - 1)]\).

- **Use the \(\otimes_{in}\)** (independence) operator when \(e_1\) and \(e_2\) are independent.
  \(([L_1, U_1] \otimes_{in} [L_2, U_2]) = [L_1 \cdot L_2, U_1 \cdot U_2]\).

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 [30].

Probabilistic conjunctions will be useful when describing TATA and TPA semantics for **cartesian products** ([5, 6, 26, 27]).

**Definition 2.13 (probabilistic disjunction strategy)** Let events \(e_1, e_2\) have probabilistic intervals \([L_1, U_1]\) and \([L_2, U_2]\) respectively. Then a **probabilistic disjunction strategy** is a binary operation \(\oplus\) which uses this information to compute the probabilistic interval \([L, U]\) for event \((e_1 \lor e_2)\). When the events involved are clear from context, we use “\([L, U] = [L_1, U_1] \oplus [L_2, U_2]\)” to denote “\((e_1 \lor e_2, [L, U]) = (e_1, [L_1, U_1]) \oplus (e_2, [L_2, U_2])\)”.

Every disjunctive strategy must conform to the following probabilistic postulates:

1. **Bottomline**: \(([L_1, U_1] \oplus [L_2, U_2]) \geq [\max(L_1, L_2), \max(U_1, U_2)]\).

2. **Ignorance**: \(([L_1, U_1] \oplus [L_2, U_2]) \subseteq [\max(L_1, L_2), \min(1, U_1 + U_2)]\). The rationale for this axiom is similar to that described for Ignorance in conjunction strategies earlier. This expression was also derived by Boole in 1854 [7].

3. **Identity**: \(([L_1, U_1] \oplus [0, 0]) = [L_1, U_1]\).

4. **Annihilator**: \(([L_1, U_1] \oplus [1, 1]) = [1, 1]\).

5. **Commutativity**: \(([L_1, U_1] \oplus [L_2, U_2]) = ([L_2, U_2] \oplus [L_1, U_1])\).
6. **Associativity:** \( ([L_1, U_1] \oplus [L_2, U_2]) \oplus [L_3, U_3]) = ([L_1, U_1] \oplus ([L_2, U_2] \oplus [L_3, U_3])) \).

7. **Monotonicity:** \( ([L_1, U_1] \oplus [L_2, U_2]) \leq ([L_1, U_1] \oplus [L_3, U_3]) \) if \([L_2, U_2] \leq [L_3, U_3] \).

The following are some sample disjunctive strategies:

- **Ignorance** \((\oplus_{ig})\) operator when we do not know the dependencies between \(e_1\) and \(e_2\).
  \( ([L_1, U_1] \oplus_{ig} [L_2, U_2]) = [\max (L_1, L_2), \min (1, U_1 + U_2)] \).

- **Positive correlation** \((\oplus_{pc})\) operator when the overlap between \(e_1\) and \(e_2\) is maximal.
  \( ([L_1, U_1] \oplus_{pc} [L_2, U_2]) = [\max (L_1, L_2), \max (U_1, U_2)] \).

- **Negative correlation** \((\oplus_{nc})\) operator when the overlap between \(e_1\) and \(e_2\) is minimal.
  \( ([L_1, U_1] \oplus_{nc} [L_2, U_2]) = [\min (1, L_1 + L_2), \min (1, U_1 + U_2)] \).

- **Independence** \((\oplus_{in})\) operator when \(e_1\) and \(e_2\) are independent.
  \( ([L_1, U_1] \oplus_{in} [L_2, U_2]) = [L_1 + L_2 - (L_1 \cdot L_2), U_1 + U_2 - (U_1 \cdot U_2)] \).

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.

### 2.6 Combination functions

Suppose that we are trying 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 \(t\). Here, combination functions can be used as a generic mechanism for combining these intervals into a single \([L, U]\) result.

**Definition 2.14 (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** \(\chi\) 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:** \(L = \max \{L_i : [L_i, U_i] \in S\}\). In other words, the lower bound of the result cannot exceed the largest lower bound of the intervals in \(S\). \(

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

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 2.16 (equity combination function)** An equity combination function $\chi$ is a combination function where $(\bigcap_{[L, U] \in S} [L, U] \neq \emptyset) \Rightarrow (\chi_i(S) = \bigcap_{[L, U] \in S} [L, U])$.

The following example shows a variety of equity combination functions.

**Example 2.5 (example equity combination functions)**:

<table>
<thead>
<tr>
<th>Name</th>
<th>Interval Returned when $\bigcap_{[L, U] \in S} [L, U] = \emptyset$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimistic Equity</td>
<td>$\chi_{eq}(S) = \max{L_i \mid [L_i, U_i] \in S}, \max{U_i \mid [L_i, U_i] \in S}$</td>
</tr>
<tr>
<td>Enclosing Equity</td>
<td>$\chi_{ec}(S) = \min{L_i \mid [L_i, U_i] \in S}, \max{U_i \mid [L_i, U_i] \in S}$</td>
</tr>
<tr>
<td>Pessimistic Equity</td>
<td>$\chi_{pe}(S) = \min{L_i \mid [L_i, U_i] \in S}, \min{U_i \mid [L_i, U_i] \in S}$</td>
</tr>
<tr>
<td>Rejecting Equity</td>
<td>$\chi_{re}(S) = [0,0]$</td>
</tr>
<tr>
<td>Skeptical Equity</td>
<td>$\chi_{ske}(S) = [0,1]$</td>
</tr>
<tr>
<td>Quasi-independence Equity</td>
<td>$\chi_{qi}(S) = \Pi_{[L_i, U_i] \in S} L_i, \Pi_{[L_i, U_i] \in S} U_i$</td>
</tr>
</tbody>
</table>

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.

#### 3.1 TP-case statements

We are now ready to define a TP-case statement and its constituent TP-cases.

**Definition 3.1 (TP-case statement over calendar $\tau$)** A TP-case statement over calendar $\tau$, denoted $\gamma$, is an expression of the form $\langle C_1, D_1, L_1, U_1, \delta_1 \rangle, \ldots, \langle C_n, D_n, L_n, U_n, \delta_n \rangle$ where $n \geq 1$, $C_i$ and $D_i$ are a temporal constraints over $\tau$, $L_i$ and $U_i$ are probabilities, $\delta_i$ is a distribution function over $\tau$, 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 $S_\tau$.
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, $(C_i \land C_j)$ 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 $(D_i \land D_j)$.
For each \(1 \leq i \leq n\), \(\gamma_i = \langle C_i, D_i, L_i, U_i, \hat{\delta}_i \rangle\) is called a TP-case of \(\gamma\). On occasion, we may want to assign probabilities to every time point in \(S_t\). Here, \(\text{sol}(C_n) = \text{sol}(\neg C_1 \land \neg C_2 \land \ldots \land \neg C_{n-1})\) and \(\gamma_n\) is called the catch-all case. For brevity, when \(\gamma_n\) 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 \(C_i\).

The reader may wonder about the occurrence of two constraints \((C_i\) and \(D_i)\) in a TP-case \(\gamma_i = \langle C_i, D_i, L_i, U_i, \hat{\delta}_i \rangle\). Intuitively, \(\text{sol}(C_i)\) is the set of time points which \(\gamma_i\) 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 \(\hat{\delta}_i\). When the above TP-case is associated with a data-tuple \(d\), it says that \(d\) is in some relation at some time point \(t\) that is a solution of the constraint \(C_i\). The probability that \(d\) is true in the relation at such a \(t\) is \(\hat{\delta}_i(D_i, t)\). In other words, the constraint \(D_i\) is used to specify the set of time points used when distributing the probability interval \([L_i, U_i]\) according to \(\hat{\delta}_i\). 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 \(\hat{\delta}_i = \{b, 0.5\}\). 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 \(S' = \{2, 3\} \subseteq S\). If we had no \(D_i\) field in our TP-cases, we could merely carry over the fact that \(S'\) has the binomial distribution on it. But applying the binomial distribution to this set 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:

- Carry with us the original set of values over which a probability distribution was defined, or
- Determine how to accurately refine 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 have chosen the first option above.

For another (simpler) example, consider a TP-case statement with one TP-case \(\gamma_1:\)
\[
\{\langle 1/8/1997 \sim 5/8/1997 \rangle, (1/8/1997 \sim 10/8/1997), 0.4, 0.8, u \rangle \}
\]
Intuitively, \(\gamma_1\) 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 \(\text{sol}(C_1)\)). Since \(\hat{\delta}_1 = \{u\}\), 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 \(\text{sol}(D_1)\). In general, the probability interval for some time point \(t \in \text{sol}(C_i)\) is \([L_i \cdot \hat{\delta}_i(D_i, t), U_i \cdot \hat{\delta}_i(D_i, t)]\). Here, we see that a TP-case \(\langle C_i, D_i, L_i, U_i, \hat{\delta}_i \rangle\) is simply an extension to the P-tuple \(\langle D_i, L_i, U_i, \hat{\delta}_i \rangle\).

Comment 3.1 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.
\[
\{((\#), (\text{month} < 6 \land \text{year} = 1997), 0.4, 0.8, g))\} \quad \{((\#), (\text{year} = 1996 \land \text{year} = 1997), 0.4, 0.8, g)), ((\#), (\text{year} = 1998), 0.0, 0.0, u)\}
\]
Furthermore,
\[
\{((\#), (\text{month} < 6 \land \text{year} = 1997), 0.4, 0.8, g)), ((\#), (\text{month} \geq 3 \land \text{year} = 1997), 0.6, 0.6, u)\}
\]
is not a TP-case statement as \((C_1 \land C_2)\) is not inconsistent (i.e., the probabilities for the overlapping time points are overspecified).

We reiterate that each temporal constraint in a TP-case statement must have a finite number of solutions (since \(S_\tau\) 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

In this section, we will define the important concept of a Temporal Probabilistic tuple (TP-tuple for short). We will require that all TP-tuples contain a special field called a “hidden field.” Intuitively, two TP-tuples with equal “data” parts may represent two different events. The hidden field (similar in function, but easier to implement than the concept of a path introduced in [30]) keeps track of how these TP-tuples were derived in order to determine whether or not two TP-tuples refer to the same event.

**Definition 3.2 (hidden field)**  A hidden field holds a lexicographically sorted hidden list of field-value pairs (i.e., \(\langle \text{field}_1, \text{value}_1 \rangle, \ldots, \langle \text{field}_n, \text{value}_n \rangle\)). If there are no pairs to store, the hidden list will be EMPTY.

In base relations, 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 \(\langle \text{field}, \text{value} \rangle\) 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.

**Definition 3.3 (TP-tuple)**  Let \(T_1 \subseteq \cdots \subseteq T_m\) be the linear hierarchy of time units over calendar \(\tau\) and suppose \(A = (A_1, \ldots, A_k)\) is a relational schema where for all \(1 \leq i < k, A_i \notin \{"C", "D", "L", "U", "\#", "L_t", "U_t", "H"\}\), 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 \(\gamma\) be a TP-case statement over \(\tau\). Then \(tp = (d, \gamma)\) is a TP-tuple over relational schema \(A\) and calendar \(\tau\). Intuitively, \(\gamma\) gives the probability for each \(t \in S_\tau\) that \(d\) occurs at time \(t\).

For instance, suppose our relational schema is \(A = (\text{Item}, \text{Origin}, \text{Dest}, \text{H})\). Then

<table>
<thead>
<tr>
<th>Item</th>
<th>Origin</th>
<th>Dest</th>
<th>H</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>I1</td>
<td>Rome</td>
<td>Vienna</td>
<td>(#)</td>
<td>day &lt; 15 &amp; month = 11 &amp; year = 1996</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(#)</td>
<td>day &gt; 15 &amp; month = 11 &amp; year = 1996</td>
<td>0.5</td>
</tr>
</tbody>
</table>

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% 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 \( I_1 \)'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 \( t \) which lies outside of November 1996 since for all \( 1 \leq i \leq n, t \notin \text{sol}(C_i) \).

Some of our definitions in the following sections will rely upon the following operators which manipulate hidden lists.

**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 \( P(d) \), is defined as \((d_1, \ldots, d_{k-1})\). In other words, the tuple \( P(d) \) contains every value in \( d \) except the hidden list \((d_k)\). Here, \( A_1 \) to \( A_{k-1} \) are known as manifest data fields.

Intuitively, the manifest projection of a TP-relation simply eliminates the hidden field of the TP-relation. The following definition specifies how hidden lists are concatenated.

**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 \( d.H = \text{"Fld3:Val3, Fld6:Val6"} \) and \( d'.H = \text{"Fld4:Val4, Fld8:Val8, Fld9:Val9"} \), then \( h'' = \text{"Fld3:Val3, Fld4:Val4, Fld6:Val6, Fld8:Val8, Fld9:Val9"} \).

Intuitively, the concatenation of two hidden lists can be obtained by taking the union of the two hidden lists, and then sorting them in lexicographic order.

### 3.3 TP-relations

We may now define a TP-relation in terms of TP-tuples.

**Definition 3.6 (TP-relation)** A TP-relation over relational schema \( A \) and calendar \( \tau \), denoted \( r \), is a multiset of TP-tuples over relational schema \( A \) and calendar \( \tau \). Intuitively, a base TP-relation is a TP-relation which did not result from a query. If \( r \) is a base TP-relation, then for each TP-tuple \( tp = (d, \gamma) \in r \), (i) \( d.H = \text{EMPTY} \) and (ii) for each TP-case \( \langle C_i, D_i, L_i, U_i, b_i \rangle \in \gamma, C_i = D_i \).

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 [25]. 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 TP-database over calendar \( \tau \) is a pair \((Base, MView)\) where \( Base \) is a set of base TP-relations over \( \tau \) and \( MView \) is a set of non-base TP-relations over \( \tau \).

For simplicity, we require all TP-relations in a TP-database to use the same calendar. Note that this requirement does not force us to lose any expressional power. Throughout this paper, we assume that all TP-relations are in the same TP-database unless we specifically state otherwise.
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 [44] to the case of TP-relations, by extending the concept of an interpretation in classical logic [44] to handle TP-relations. Before doing this, a preliminary definition is needed.

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

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 \( t_p \) and \( t_p' \) are data-identical, we assume that they provide complementary information for the same event. If \( t_p \) and \( t_p' \) are not data-identical, we assume that they refer to different events. Let \( t_p \in r \). Then \( r[t_p] \) denotes the multiset of all TP-tuples in \( r \) which are data-identical to \( t_p \). Since “data-identical” is a reflexive, symmetric, transitive relation on TP-tuples, it is also an equivalence relation on \( r \) where each \( r[t_p] \) 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 \( t_p = (d, \gamma) \in r \) where \( t \in \text{sol}(C_1 \lor \cdots \lor C_n) \). Otherwise, since \( r \) 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 \( r[t_p] \) corresponds to an equivalence class in this relation.

Intuitively, a TP-tuple \( t_p = (d, \gamma) \) 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 \( \tau \) be a calendar, and let \( \text{dom}(A) = \text{dom}(A_1) \times \cdots \times \text{dom}(A_k) \) be the domain of \( A \). Then a TP-interpretation over the pair \( A, \tau \) is a function \( I_{A, \tau} : \text{dom}(A) \rightarrow [0, 1] \) such that \( (\forall d \in \text{dom}(A))(\sum_{t \in \text{sol}(D)} I_{A, \tau}(d, t) \leq 1) \).

Let \( \epsilon \) 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 \( \epsilon \) is true at time point \( t \) is \( p \). Let \( D \) be a temporal constraint over \( \tau \). Then the probability assigned by \( I_{A, \tau} \) to \( D \), denoted \( I_{A, \tau}(d, D) \), is equal to \( \sum_{t \in \text{sol}(D)} I_{A, \tau}(d, t) \). 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. 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. \( L_i \leq I_{A, \tau}(d, D_i) \leq U_i \), i.e. the probability that \( I_{A, \tau} \) assigns to \( D_i \) lies in the interval \([L_i, U_i]\).
2. \( (\forall t \in \text{sol}(C_i))(I_{A, \tau}(d, D_i) \cdot \delta_i(D_i, t) = I_{A, \tau}(d, t)) \), i.e. \( I_{A, \tau} \) distributes probabilities for each \( t \in \text{sol}(C_i) \) according to \( \delta_i \).

TP-interpretation \( I_{A, \tau} \) satisfies TP-tuple \( t_p = (d, \gamma) \), denoted \( I_{A, \tau} \models t_p \), 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.

<table>
<thead>
<tr>
<th>Item</th>
<th>Origin</th>
<th>Dest</th>
<th>H</th>
<th>C</th>
<th>D</th>
<th>L</th>
<th>U</th>
<th>$\delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>Rome</td>
<td>Vienna</td>
<td>(1)</td>
<td>(1)</td>
<td>$\text{day &lt; 15 \land month = 11 \land year = 1996}$</td>
<td>0.5</td>
<td>0.6</td>
<td>$u$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(1)</td>
<td>(1)</td>
<td>$\text{day \geq 15 \land month = 11 \land year = 1996}$</td>
<td>0.4</td>
<td>0.4</td>
<td>$u$</td>
</tr>
</tbody>
</table>
Consider the TP-interpretation defined as follows:

\[
I(\langle \text{II}, \text{Rome}, \text{Vienna} \rangle, (d, 11, 1996)) = 0.04 \text{ when } d < 15.
\]

\[
I(\langle \text{II}, \text{Rome}, \text{Vienna} \rangle, (d, 11, 1996)) = \frac{0.4}{16} \text{ when } d \geq 15.
\]

\[
I(\langle \text{item}, \text{origin}, \text{dest}, (d, m, y) \rangle) = 0 \text{ otherwise.}
\]

This TP-interpretation satisfies the TP-tuple above because

\[
I(\langle \text{II}, \text{Rome}, \text{Vienna} \rangle, \{t \mid t.day < 15 \land t.month = 11 \land t.year = 1996\}) = 0.04 \times 14 = 0.56
\]

which lies between 0.5 and 0.6.

**Definition 3.11 (consistency and mutual consistency)** A TP-tuple \( tp \) is consistent iff there exists a TP-interpretation \( I_{A,T} \) where \( I_{A,T} \models tp \). A TP-relation \( r \) is consistent iff \( (\exists I_{A,T})(\forall tp \in r)(I_{A,T} \models tp) \). TP-relations \( r \) and \( r' \) are mutually consistent iff \( (\exists I_{A,T})(\forall tp \in r)(I_{A,T} \models tp) \land (\forall tp' \in r')(I_{A,T} \models tp') \). Note that if consistent TP-relations \( r, r' \) have different schemas, then \( r \) and \( r' \) must be mutually consistent. \( \Box \)

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

\[
(L_1 + \cdots + L_n) \leq \min(1, \sum_{i=1}^{n} L_i \cdot \sum_{t \in S_r} \delta_i(D_i, t)).
\]

If this condition holds, then the TP-tuple is consistent. This forms the basis for the following claim. When the distribution function used is determinate, then the quantity on the right side of the above inequality can be computed in linear time because the determinacy condition guarantees constant time computation of the sum \( \sum_{t \in S_r} \delta_i(D_i, t) \).

**Proposition 3** Checking consistency of a compact TP-relation which uses determinate PDFs is linear in the size of the TP-relation.

### 4 Annotated relations

Annotated relations are the flat, relational equivalents of TP-relations. Let TP-case statement \( \gamma = \{\langle C_1, D_1, L_1, U_1, \delta_1 \rangle, \ldots, \langle C_n, D_n, L_n, U_n, \delta_n \rangle\} \). Then we can “flatten” TP-tuple \( tp = (d, \gamma) \) by creating an annotated tuple for each time point \( t \in \text{sol}(C_i) \cup \cdots \cup \text{sol}(C_n) \).

Each annotated tuple \((at)\) provides probabilistic information \([L_i, U_i]\) for one data tuple \((d)\) at one point in time \((t)\). According to the definition for a TP-case statement, \( t \in \text{sol}(C_i) \) for at most one \( \gamma_i = \langle C_i, D_i, L_i, U_i, \delta_i \rangle \in \gamma \) since \((i \neq j) \Rightarrow (\text{sol}(C_i) \cap \text{sol}(C_j) = \emptyset) \). If \( t \notin \bigcup_{i \in \gamma} \text{sol}(C_i) \), then we do not create an annotated tuple for \( t \). We denote the set of all annotated tuples derived from \( tp \) by \( \text{ANN}(tp) \). In the worst case, \( \text{ANN}(tp) \) may contain \(|S_r| \) tuples.

Let \( r \) be a TP-relation containing \( n \) TP-tuples and let \( \sqcup \) denote the multiset union operator (in other words, a union operation without duplicate elimination). Then the annotated relation for \( r \), denoted \( \text{ANN}(r) \), is defined as \( \sqcup_{tp \in r} \text{ANN}(tp) \). This means that \( \text{ANN}(r) \) may contain up to \( n \cdot |S_r| \) tuples! In general, annotated
relations will always be finite since they are derived from a finite number of TP-cases, and each TP-case pertains to a finite number of time points (since \( \tau \) is finite). Nonetheless, we can clearly see that \( \text{ANN}(t_p) \) and \( \text{ANN}(\tau) \) will often be large and impractical. 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.

Let \( \tau \) consist of one TP-tuple which contains data tuple (“D1”, EMPTY) and TP-case \( \gamma_1 \) as shown below.

<table>
<thead>
<tr>
<th>Data</th>
<th>H</th>
<th>C</th>
<th>D</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>(#)</td>
<td>day ( \leq 4 ) &amp; month = 11 &amp; year = 1996</td>
<td>0.4</td>
<td>0.8</td>
</tr>
</tbody>
</table>

Then \( \text{ANN}(\tau) \) will be

<table>
<thead>
<tr>
<th>Data</th>
<th>H</th>
<th>Day</th>
<th>Month</th>
<th>Year</th>
<th>L</th>
<th>U</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>1</td>
<td>11</td>
<td>1996</td>
<td>0.1</td>
<td>0.2</td>
<td></td>
</tr>
<tr>
<td>D1</td>
<td>2</td>
<td>11</td>
<td>1996</td>
<td>0.1</td>
<td>0.2</td>
<td></td>
</tr>
<tr>
<td>D1</td>
<td>3</td>
<td>11</td>
<td>1996</td>
<td>0.1</td>
<td>0.2</td>
<td></td>
</tr>
<tr>
<td>D1</td>
<td>4</td>
<td>11</td>
<td>1996</td>
<td>0.1</td>
<td>0.2</td>
<td></td>
</tr>
</tbody>
</table>

However if \( \gamma_1 \)’s \( C_1 \) field was “day \( \leq 3 \) \& month = 11 \& year = 1996”, then \( \text{ANN}(\tau) \) would no longer contain the last tuple shown above. In general, note that changing \( C_i \) only affects the number of annotated tuples in \( \text{ANN}(\tau) \), 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 \( \text{ANN}(\tau) \). We were only justified in making this uniformity assumption since \( \delta_i = \text{“a”} \). 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 \( \delta_i \).

In this section, we first present a more formal definition for annotated relations. We then give an example to show how annotated relations change when we vary the distribution functions.

### 4.1 Formal definitions

**Definition 4.1 (annotated relation for a TP-tuple)** Let \( t_p = (d, \gamma) \) be a TP-tuple over relational schema \( (A_1, \ldots, A_k) \) and calendar \( \tau \) where \( d = (d_1, \ldots, d_k) \). Suppose \( \gamma \) 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 \( \tau \) consists of a linear hierarchy \( H \) containing \( m \) time units \( T_1 \supseteq \cdots \supseteq T_m \). Here, each \( t \in S_\tau \) will be of the form \( t = (v_1, \ldots, v_m) \).

Then the **annotated relation for TP-tuple** \( t_p \) **over calendar** \( \tau \), denoted \( \text{ANN}(t_p) \), is defined as 

\[
\{ (d, t, L_i, U_i) \mid t \in \text{sol}(C_i) \text{ for some } \gamma_i \in \gamma \text{ and } [L_i, U_i] = [L_i \cdot x, U_i \cdot x] \text{ where } x = \delta_i(D_i, t) \}\n\]

Intuitively, in the definition above, \( x \) represents the percentage of \( \text{sol}(D_i) \)’s probability which is associated with time point \( t \) according to \( \delta_i \). Note that when we explicitly show all fields of an annotated tuple, \( at = (d_1, \ldots, d_k, v_1, \ldots, v_m, L_i, U_i) \) is over the schema \((A_1, \ldots, A_k, T_1, \ldots, T_m, L_i, U_i)\). 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 \( L_i, U_i \) are **probabilistic fields**.

**Definition 4.2 (annotated relation for a TP-relation)** Let \( \tau \) be a TP-relation over \( \tau \) containing \( n \) TP-tuples \( t_{p_1} \ldots t_{p_n} \). Then the **annotated relation for TP-relation** \( \tau \) **over calendar** \( \tau \), denoted \( \text{ANN}(\tau) \), is defined as the multiset \( \text{ANN}(t_{p_1}) \uplus \ldots \uplus \text{ANN}(t_{p_n}) \) over \( \tau \).
We associate with each ANN(r) the primary key which is associated with r. This key will be used when we describe the TATA’s projection operation (§5.7).

4.2 Semantics and consistency of annotated relations

Our semantics for annotated relations closely parallels our semantics for TP-relations (§3.4).

**Definition 4.3 (data-identical annotated tuples)** Annotated tuples \( at = (d, t, I_t, U_t) \) and \( at' = (d', t', I'_t, U'_t) \) are 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 that they refer to different events. Let \( d \) be a data tuple and let \( t \) be a time point. Then \( \text{ANN}(r)[d, t] \) denotes the equivalence class of the pair \((d, t)\), i.e., the multiset of all \( at \in \text{ANN}(r) \) where \( (at.d = d \land at.t = t) \).

Suppose \( at = (d, t, I_t, U_t) \in \text{ANN}(r) \), \( at' = (d', t', I'_t, U'_t) \in \text{ANN}(r) \), and \((d = d' \land t = t') \). Here, since \( at, at' \in \text{ANN}(r) \) refer to the same event at the same point in time, \( \text{ANN}(r) \) is an uncompact annotated relation. If there are no pairs of annotated tuples \( at, at' \in \text{ANN}(r) \) where \( (d = d' \land t = t') \), then \( \text{ANN}(r) \) 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, \( \text{ANN}(r) \), is compact.

We may now define what is means for an annotated relation to be satisfied by a TP-interpretation.

**Definition 4.4 (satisfaction of annotated tuples)** Let \( d \) be a tuple in relational schema \( A \), let \( t \) be a time point in \( S_t \), and let \([L, U]\) be a probability interval. Then a TP-interpretation \( I_{A,\tau} \) satisfies annotated tuple \( at = (d, t, I_t, U_t) \), denoted \( I_{A,\tau} \models at \), iff \( L_t \leq I_{A,\tau}(d, t) \leq U_t \).

**Definition 4.5 (consistency of annotated relations)** An annotated tuple \( at \) is consistent iff \( \exists I_{A,\tau} \models I_{A,\tau} \models at \). An annotated relation \( \text{ANN}(r) \) is consistent iff \( \exists I_{A,\tau} \models \forall at \in \text{ANN}(r) \models I_{A,\tau} \models at \). Annotated relations \( \text{ANN}(r) \) and \( \text{ANN}(r') \) are mutually consistent iff \( \exists I_{A,\tau} \models \forall at \in \text{ANN}(r) \models I_{A,\tau} \models at \land \forall at' \in \text{ANN}(r') \models I_{A,\tau} \models at' \).

The following theorem tells us that if \( r \) is a consistent TP-relation, then \( \text{ANN}(r) \) is also consistent.

**Theorem 2** Let \( r \) be a TP-relation. If \( I_{A,\tau} \) satisfies \( r \), then \( I_{A,\tau} \) also satisfies \( \text{ANN}(r) \). Hence if \( r \) is consistent, so is \( \text{ANN}(r) \).

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

**Example 4.1 (satisfaction)** Let \( r \) consist of one TP-tuple \((d, \gamma)\) where \( \gamma = \{\langle \#, 1 \sim 2 \rangle, 0.4, 0.8, u\} \). Then \( \text{ANN}(r) = \{at_1, at_2\} \) 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 \( \text{ANN}(r) \), but \( I_{A,\tau} \) does not satisfy \( r \) because every TP-interpretation \( J_{A,\tau} \) that satisfies \( r \) must have \( J_{A,\tau}(d, 1) = J_{A,\tau}(d, 2) \).
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, if \((d, \gamma) \in r\), and if \((d, t, L_t, U_t) \in \text{ANN}(r)\), then there must be a TP-interpretation \( I_{A,r} \) of \( r \) such that \( I_{A,r}(d, t) = L_t \). 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 \( t \).

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

Theorems 2 and 3 jointly tells 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 compact a TP-relation \( r \).

Note that the above result 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:

<table>
<thead>
<tr>
<th>Data</th>
<th>H</th>
<th>C</th>
<th>D</th>
<th>L</th>
<th>U</th>
<th>( \delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td></td>
<td></td>
<td></td>
<td>0.6</td>
<td>1</td>
<td>u</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.4</td>
<td>1</td>
<td>u</td>
</tr>
</tbody>
</table>

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 4.6 (tightening)** Let \( \gamma = \{\gamma_1, \ldots, \gamma_n\} \) be a TP-case statement where each TP-case \( \gamma_i = \langle C_i, D_i, L_i, U_i, b_i \rangle \). A tightening of \( \gamma \) returns a TP-case statement \( \gamma'' = \{\gamma''_1, \ldots, \gamma''_n\} \) where each \( \gamma''_i = \langle C''_i, D''_i, L''_i, U''_i, b''_i \rangle \) and each \( U''_i \leq U_i \) for all \( 1 \leq i \leq n \).

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

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

The following theorem tells us that for tight and compact TP-relations, Theorem 3 holds for upper bounds as well.

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

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. Later, in Section 6.10, we will describe a procedure for tightening TP-relations.

**4.3 Sample annotated relations**

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

<table>
<thead>
<tr>
<th>Item</th>
<th>Origin</th>
<th>Dest</th>
<th>H</th>
<th>C</th>
<th>D</th>
<th>L</th>
<th>U</th>
<th>( \delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>I1</td>
<td>Rome</td>
<td>Paris</td>
<td></td>
<td></td>
<td>(\text{day } \leq 2 \land \text{month } = 8 \land \text{year } = 1997)</td>
<td>0.5</td>
<td>0.7</td>
<td>(\phi)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(\text{day } \geq 5 \land \text{day } \leq 7 \land \text{month } = 8 \land \text{year } = 1997)</td>
<td>0.3</td>
<td>0.6</td>
<td>(\phi)</td>
</tr>
</tbody>
</table>
Note that the variable $\Phi$ must be instantiated. If $\Phi = \text{"u"}$, $\text{ANN}(r)$ will be

<table>
<thead>
<tr>
<th>Item</th>
<th>Origin</th>
<th>Dest</th>
<th>$H$</th>
<th>Day</th>
<th>Month</th>
<th>Year</th>
<th>$L_t$</th>
<th>$U_t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Rome</td>
<td>Paris</td>
<td>1</td>
<td>8</td>
<td>1997</td>
<td>0.25</td>
<td>0.35</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Rome</td>
<td>Paris</td>
<td>2</td>
<td>8</td>
<td>1997</td>
<td>0.25</td>
<td>0.35</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Rome</td>
<td>Paris</td>
<td>5</td>
<td>8</td>
<td>1997</td>
<td>0.10</td>
<td>0.20</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Rome</td>
<td>Paris</td>
<td>6</td>
<td>8</td>
<td>1997</td>
<td>0.10</td>
<td>0.20</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Rome</td>
<td>Paris</td>
<td>7</td>
<td>8</td>
<td>1997</td>
<td>0.10</td>
<td>0.20</td>
<td></td>
</tr>
</tbody>
</table>

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 $\text{ANN}(r)$. However if $\Phi = \text{"g"}$, $\text{ANN}(r)$ will be

<table>
<thead>
<tr>
<th>Item</th>
<th>Origin</th>
<th>Dest</th>
<th>$H$</th>
<th>Day</th>
<th>Month</th>
<th>Year</th>
<th>$L_t$</th>
<th>$U_t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Rome</td>
<td>Paris</td>
<td>1</td>
<td>8</td>
<td>1997</td>
<td>0.25</td>
<td>0.35</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Rome</td>
<td>Paris</td>
<td>2</td>
<td>8</td>
<td>1997</td>
<td>0.125</td>
<td>0.175</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Rome</td>
<td>Paris</td>
<td>5</td>
<td>8</td>
<td>1997</td>
<td>0.10</td>
<td>0.20</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Rome</td>
<td>Paris</td>
<td>6</td>
<td>8</td>
<td>1997</td>
<td>0.10</td>
<td>0.20</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Rome</td>
<td>Paris</td>
<td>7</td>
<td>8</td>
<td>1997</td>
<td>0.10</td>
<td>0.20</td>
<td></td>
</tr>
</tbody>
</table>

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

## 5 Theoretical Annotated Temporal Algebra

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

Figure 2 shows what we hope to accomplish through this section. 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 will 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 implement the annotated algebra operations on the implicit (smaller) TP-relations, rather than their larger annotated counterparts.
The definitions in this section will produce a new annotated relation \( \text{ANN}(r^p) \) based on input from consistent annotated relations \( \text{ANN}(r), \text{ANN}(r^p) \). Oftentimes, these definitions will refer to annotated tuples \( at, at' \) which are assumed to be of the form \( at = (d, t, L_i, U_i) \) and \( at' = (d', t', L'_i, U'_i) \).

**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 \( \kappa \) from annotated relations to annotated relations is called a compaction operation if it satisfies the following axioms:

- **Compactness**: \( \kappa(\text{ANN}(r)) \) is compact for all annotated relations \( \text{ANN}(r) \).

- **No Fooling Around (NFA)**: If \( \text{ANN}(r) \) is compact, then \( \kappa(\text{ANN}(r)) = \text{ANN}(r) \).

- **Conservativeness**: If \( at = (d, t, L_i, U_i) \in \kappa(\text{ANN}(r)) \), then \( \exists at' = (d, t, L'_i, U'_i) \in \text{ANN}(r) \). □

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 axiom jointly guarantee that compaction operations are idempotent, i.e. \( \kappa(\kappa(\text{ANN}(r))) = \kappa(\text{ANN}(r)) \). 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 2.6).

**Definition 5.2 (\( \chi \)-compaction of an annotated relation)** Let \( \chi \) be a combination function. Then the \( \chi \)-compaction of annotated relation \( \text{ANN}(r) \), denoted \( \kappa_\chi(\text{ANN}(r)) \), is defined as \( \kappa_\chi(\text{ANN}(r)) = \{at = (d, t, L_i, U_i) \mid [L_i, U_i] = \chi(\{[L_1^{(d,t)}, U_1^{(d,t)}], \ldots, [L_k^{(d,t)}, U_k^{(d,t)}]\})\} \) where \( \text{ANN}(r)[d, t] = \{at_1^{(d,t)}, \ldots, at_k^{(d,t)}\} \) and \( at_i^{(d,t)} = (d, t, L_i^{(d,t)}, U_i^{(d,t)}) \). □

Intuitively, in combination function based compactions, \( \chi \) 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.

**Proposition 4** Let \( \chi \) be any combination function. Then \( \kappa_\chi(\text{ANN}(r)) \) is a compaction operation.

Theorem 5 indicates that \( \kappa_\chi(\text{ANN}(r)) \) operations also possess another important property: Reasonableness. Intuitively, this property is similar to the converse of Conservativeness — every tuple in \( \text{ANN}(r) \) leads to a corresponding tuple in the result of the compaction.

**Theorem 5** If \( at' = (d, t, L'_i, U'_i) \in \text{ANN}(r) \), then \( \exists at = (d, t, L_i, U_i) \in \kappa_\chi(\text{ANN}(r)) \).

23
### Figure 3: Example Base TP and Annotated Relations

#### r1

<table>
<thead>
<tr>
<th>Data</th>
<th>H</th>
<th>C</th>
<th>D</th>
<th>L</th>
<th>U</th>
<th>δ</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td></td>
<td>(#) (1/8/1997 ~ 3/8/1997)</td>
<td>(1/8/1997 ~ 3/8/1997)</td>
<td>0.64</td>
<td>0.88</td>
<td>g</td>
</tr>
</tbody>
</table>

#### ANN(r1)

<table>
<thead>
<tr>
<th>Data</th>
<th>H</th>
<th>Day</th>
<th>Month</th>
<th>Year</th>
<th>L₂</th>
<th>U₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>1</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.32</td>
<td>0.44</td>
</tr>
<tr>
<td>D1</td>
<td>2</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.16</td>
<td>0.22</td>
</tr>
<tr>
<td>D1</td>
<td>3</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.08</td>
<td>0.11</td>
</tr>
<tr>
<td>D1</td>
<td>5</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.10</td>
<td>0.20</td>
</tr>
<tr>
<td>D1</td>
<td>6</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.10</td>
<td>0.20</td>
</tr>
<tr>
<td>D1</td>
<td>7</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.10</td>
<td>0.20</td>
</tr>
<tr>
<td>D1</td>
<td>8</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.10</td>
<td>0.20</td>
</tr>
</tbody>
</table>

#### r2

<table>
<thead>
<tr>
<th>Data</th>
<th>H</th>
<th>C</th>
<th>D</th>
<th>L</th>
<th>U</th>
<th>δ</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td></td>
<td>(#) (2/8/1997 ~ 3/8/1997)</td>
<td>(2/8/1997 ~ 3/8/1997)</td>
<td>0.20</td>
<td>0.50</td>
<td>g</td>
</tr>
</tbody>
</table>

#### ANN(r2)

<table>
<thead>
<tr>
<th>Data</th>
<th>H</th>
<th>Day</th>
<th>Month</th>
<th>Year</th>
<th>L₂</th>
<th>U₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>2</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.10</td>
<td>0.25</td>
</tr>
<tr>
<td>D1</td>
<td>3</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.05</td>
<td>0.125</td>
</tr>
<tr>
<td>D1</td>
<td>6</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.10</td>
<td>0.20</td>
</tr>
<tr>
<td>D1</td>
<td>7</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.10</td>
<td>0.20</td>
</tr>
<tr>
<td>D1</td>
<td>8</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.10</td>
<td>0.20</td>
</tr>
</tbody>
</table>

#### r3

<table>
<thead>
<tr>
<th>Data1</th>
<th>Data2</th>
<th>H</th>
<th>C</th>
<th>D</th>
<th>L</th>
<th>U</th>
<th>δ</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>D3</td>
<td></td>
<td>(#) (2/8/1997 ~ 3/8/1997)</td>
<td>(2/8/1997 ~ 3/8/1997)</td>
<td>0.60</td>
<td>0.80</td>
<td>g</td>
</tr>
</tbody>
</table>

#### ANN(r3)

<table>
<thead>
<tr>
<th>Data1</th>
<th>Data2</th>
<th>H</th>
<th>Day</th>
<th>Month</th>
<th>Year</th>
<th>L₂</th>
<th>U₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>D2</td>
<td></td>
<td>2</td>
<td>8</td>
<td>1997</td>
<td>0.20</td>
<td>0.40</td>
</tr>
<tr>
<td>D1</td>
<td>D3</td>
<td></td>
<td>2</td>
<td>8</td>
<td>1997</td>
<td>0.30</td>
<td>0.40</td>
</tr>
<tr>
<td>D1</td>
<td>D3</td>
<td></td>
<td>3</td>
<td>8</td>
<td>1997</td>
<td>0.15</td>
<td>0.20</td>
</tr>
</tbody>
</table>

#### r4

<table>
<thead>
<tr>
<th>Data1</th>
<th>Data2</th>
<th>H</th>
<th>C</th>
<th>D</th>
<th>L</th>
<th>U</th>
<th>δ</th>
</tr>
</thead>
<tbody>
<tr>
<td>D4</td>
<td>D5</td>
<td></td>
<td>(#) (1/8/1997 ~ 1/8/1997)</td>
<td>(1/8/1997 ~ 1/8/1997)</td>
<td>0.70</td>
<td>0.80</td>
<td>u</td>
</tr>
</tbody>
</table>

#### ANN(r4)

<table>
<thead>
<tr>
<th>Data1</th>
<th>Data2</th>
<th>H</th>
<th>Day</th>
<th>Month</th>
<th>Year</th>
<th>L₂</th>
<th>U₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>D2</td>
<td></td>
<td>2</td>
<td>8</td>
<td>1997</td>
<td>0.10</td>
<td>0.50</td>
</tr>
<tr>
<td>D1</td>
<td>D2</td>
<td></td>
<td>3</td>
<td>8</td>
<td>1997</td>
<td>0.05</td>
<td>0.25</td>
</tr>
<tr>
<td>D1</td>
<td>D3</td>
<td></td>
<td>3</td>
<td>8</td>
<td>1997</td>
<td>0.50</td>
<td>0.50</td>
</tr>
<tr>
<td>D4</td>
<td>D5</td>
<td></td>
<td>1</td>
<td>8</td>
<td>1997</td>
<td>0.70</td>
<td>0.80</td>
</tr>
</tbody>
</table>
Another possible class of compaction strategies involves the use of a $p$-strategy $\rho$ (i.e., a probabilistic conjunction or disjunction strategy as defined in Section 2.5). These compactions, denoted $\kappa_\rho$, are defined in the same way as $\kappa_\chi(\text{ANN}(r))$ except we let $[L_t, U_t] = ([L_1^{(d,t)}, U_1^{(d,t)}] \otimes_\rho \ldots \otimes_\rho [L_k^{(d,t)}, U_k^{(d,t)}])$ when $\rho$ is a conjunctive $p$-strategy, and let $[L_t, U_t] = ([L_1^{(d,t)}, U_1^{(d,t)}] \oplus_\rho \ldots \oplus_\rho [L_k^{(d,t)}, U_k^{(d,t)}])$ when $\rho$ is a conjunctive $p$-strategy.

**Proposition 5** Let $\rho$ be any (conjunctive or disjunctive) $p$-strategy. Then $\kappa_\rho(\text{ANN}(r))$ is a compaction operation.

### 5.2 Intersection of two annotated relations

The intersection of annotated relations $\text{ANN}(r)$ and $\text{ANN}(r')$ is viewed as the operation of extracting information which is common to both relations. In our algebra, we break intersection into two suboperations: First, a *multiset intersection* will extract all tuples from both $\text{ANN}(r)$ and $\text{ANN}(r')$ 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* $\text{ANN}(r)$ and $\text{ANN}(r')$, denoted $\text{ANN}(r) \cap \text{ANN}(r')$, is defined as $\text{ANN}(r'') = \{at \in \text{ANN}(r') \mid (\exists at' \in \text{ANN}(r')) (d = d' \land t = t') \cup \{at' \in \text{ANN}(r') \mid (\exists at \in \text{ANN}(r)) (d = d' \land t = t')\}.

Intuitively, $\text{ANN}(r'')$ will contain all $at \in \text{ANN}(r)$ and all $at' \in \text{ANN}(r')$ where $at$ and $at'$ refer to the same event at the same point in time. Recall that “$(\exists at \in \text{ANN}(r))” and “$(\exists at' \in \text{ANN}(r'))” are shorthand for “$(\exists (d,t, L_t, U_t) \in \text{ANN}(r))” and “$(\exists (d',t', L'_t, U'_t) \in \text{ANN}(r'))” respectively. For greater clarity and conciseness, our definitions will make use of this implicit notation. For example, $\text{ANN}(r'') = \text{ANN}(r_1) \cap \text{ANN}(r_2)$ will be

<table>
<thead>
<tr>
<th>Data</th>
<th>H</th>
<th>Day</th>
<th>Month</th>
<th>Year</th>
<th>$L_t$</th>
<th>$U_t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>2</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.16</td>
<td>0.22</td>
</tr>
<tr>
<td>D1</td>
<td>3</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.08</td>
<td>0.11</td>
</tr>
<tr>
<td>D1</td>
<td>6</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.10</td>
<td>0.20</td>
</tr>
<tr>
<td>D1</td>
<td>7</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.10</td>
<td>0.20</td>
</tr>
<tr>
<td>D1</td>
<td>8</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.10</td>
<td>0.20</td>
</tr>
<tr>
<td>D1</td>
<td>2</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.10</td>
<td>0.25</td>
</tr>
<tr>
<td>D1</td>
<td>3</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.05</td>
<td>0.125</td>
</tr>
<tr>
<td>D1</td>
<td>6</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.10</td>
<td>0.20</td>
</tr>
<tr>
<td>D1</td>
<td>7</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.10</td>
<td>0.20</td>
</tr>
<tr>
<td>D1</td>
<td>8</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.10</td>
<td>0.20</td>
</tr>
</tbody>
</table>

Clearly, $\text{ANN}(r'')$ above is uncompactred. To obtain a compact annotated relation, we may use any $\kappa_\chi$ compaction operator. Using a $\kappa_\chi$ compaction operator when defining intersection makes sense because the two different relations $r$ and $r'$ may both contain data tuple $d$ at some time point $t$, but with different probabilities. In this case, using a conjunction strategy is not appropriate because we are not combining probabilities of different events — we are combining two different probabilities assigned to the same even by two different sources (relations $r$ and $r'$). This is exactly what combination functions $\chi$ were designed to support.
Definition 5.4 (intersection of two annotated relations) The intersection of annotated relations $\text{ANN}(r)$ and $\text{ANN}(r')$ under the $\chi$ combination function, denoted $\text{ANN}(r) \cap_\chi \text{ANN}(r')$, is defined as $\kappa_\chi(\text{ANN}(r) \cap \text{ANN}(r'))$. \hfill $\Box$

For example, $\text{ANN}(r_1) \cap_{eq} \text{ANN}(r_2) = \kappa_{eq}(\text{ANN}(r''))$ will be

<table>
<thead>
<tr>
<th>Data</th>
<th>H</th>
<th>Day</th>
<th>Month</th>
<th>Year</th>
<th>$L_t$</th>
<th>$U_t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>2</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.16</td>
<td>0.22</td>
</tr>
<tr>
<td>D1</td>
<td>3</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.08</td>
<td>0.11</td>
</tr>
<tr>
<td>D1</td>
<td>6</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.10</td>
<td>0.20</td>
</tr>
<tr>
<td>D1</td>
<td>7</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.10</td>
<td>0.20</td>
</tr>
<tr>
<td>D1</td>
<td>8</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.10</td>
<td>0.20</td>
</tr>
</tbody>
</table>

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 multiset union of annotated relations $\text{ANN}(r)$ and $\text{ANN}(r')$, denoted $\text{ANN}(r) \cup \text{ANN}(r')$, is defined as $\text{ANN}(r'') = \text{ANN}(r) \cup \text{ANN}(r')$. \hfill $\Box$

Intuitively, $\text{ANN}(r'')$ will contain all $at \in \text{ANN}(r)$ and all $at' \in \text{ANN}(r')$. For example, $\text{ANN}(r'') = \text{ANN}(r_1) \cup \text{ANN}(r_2)$ will be

<table>
<thead>
<tr>
<th>Data</th>
<th>H</th>
<th>Day</th>
<th>Month</th>
<th>Year</th>
<th>$L_t$</th>
<th>$U_t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>1</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.32</td>
<td>0.44</td>
</tr>
<tr>
<td>D1</td>
<td>2</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.16</td>
<td>0.22</td>
</tr>
<tr>
<td>D1</td>
<td>3</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.08</td>
<td>0.11</td>
</tr>
<tr>
<td>D1</td>
<td>5</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.10</td>
<td>0.20</td>
</tr>
<tr>
<td>D1</td>
<td>6</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.10</td>
<td>0.20</td>
</tr>
<tr>
<td>D1</td>
<td>7</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.10</td>
<td>0.20</td>
</tr>
<tr>
<td>D1</td>
<td>8</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.10</td>
<td>0.20</td>
</tr>
<tr>
<td>D1</td>
<td>2</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.10</td>
<td>0.25</td>
</tr>
<tr>
<td>D1</td>
<td>3</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.05</td>
<td>0.125</td>
</tr>
<tr>
<td>D1</td>
<td>6</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.10</td>
<td>0.20</td>
</tr>
<tr>
<td>D1</td>
<td>7</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.10</td>
<td>0.20</td>
</tr>
<tr>
<td>D1</td>
<td>8</td>
<td>8</td>
<td>1997</td>
<td></td>
<td>0.10</td>
<td>0.20</td>
</tr>
</tbody>
</table>

As in the case of intersection, $\text{ANN}(r'')$ may overspecify probabilistic information. We can consolidate this information by using a $\kappa_\chi$ compaction operator. The reason for using the operator $\kappa_\chi$ instead of a conjunction strategy is exactly for the same reason that we used the $\kappa_\chi$ compaction operator when defining intersection (see discussion preceding Definition 5.4).

Definition 5.6 (union of two annotated relations) The union of annotated relations $\text{ANN}(r)$ and $\text{ANN}(r')$ under the $\chi$ combination function, denoted $\text{ANN}(r) \cup_\chi \text{ANN}(r')$, is defined as $\kappa_\chi(\text{ANN}(r) \cup \text{ANN}(r'))$. \hfill $\Box$

For example, $\text{ANN}(r_1) \cup_{eq} \text{ANN}(r_2) = \kappa_{eq}(\text{ANN}(r''))$ will be
Note that although $\text{ANN}(r_1)$ and $\text{ANN}(r_2)$ are both consistent, $\text{ANN}(r_1) \cup_{cq} \text{ANN}(r_2)$ above is an inconsistent annotated relation (since for some data tuple $d$, the sum of the $L_t$ values exceeds 1.0). This occurs since $\text{ANN}(r_1)$ and $\text{ANN}(r_2)$ are not mutually consistent (§4.2). In general, if consistent annotated relations $\text{ANN}(r)$ and $\text{ANN}(r')$ are also mutually consistent, then $\text{ANN}(r) \cup_{cq} \text{ANN}(r')$ will always be consistent.

### 5.4 Selection on an annotated relation

We represent a selection condition over calendar $\tau$ by the symbol $\mathcal{C}$. If $\mathcal{C}$ is of the form $(F \ \text{op} \ v)$ or $(t_1 \sim t_2)$, then $\mathcal{C}$ is an atomic condition over $\tau$. Let $\mathcal{C}$ be an atomic condition, let $T_1 \subseteq \cdots \subseteq T_m$ be a linear hierarchy $H$ of time units over $\tau$, 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 $\text{ANN}(r)$, denoted $\sigma_{\mathcal{C}}(\text{ANN}(r))$, is defined in the following way:

- If $\mathcal{C}$ is a data condition, $\text{ANN}(r^\mathcal{C}) = \{at \in \text{ANN}(r) \mid d \text{ satisfies } \mathcal{C}\}$.
  In this case, our selection is based on the classical relational algebra.
- If $\mathcal{C}$ is a temporal condition, $\text{ANN}(r^\mathcal{C}) = \{at \in \text{ANN}(r) \mid t \in \text{sol}(\mathcal{C})\}$.
- If $\mathcal{C}$ is a probabilistic condition, $\text{ANN}(r^\mathcal{C}) = \{at \in \text{ANN}(r) \mid ([L, U] = [L_t, U_t]) \text{ satisfies } \mathcal{C}\}$. □

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

<table>
<thead>
<tr>
<th>Data</th>
<th>H</th>
<th>Day</th>
<th>Month</th>
<th>Year</th>
<th>( L_t )</th>
<th>( U_t )</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>1</td>
<td>8</td>
<td>1997</td>
<td>0.32</td>
<td>0.44</td>
<td></td>
</tr>
<tr>
<td>D1</td>
<td>2</td>
<td>8</td>
<td>1997</td>
<td>0.16</td>
<td>0.22</td>
<td></td>
</tr>
<tr>
<td>D1</td>
<td>3</td>
<td>8</td>
<td>1997</td>
<td>0.08</td>
<td>0.11</td>
<td></td>
</tr>
</tbody>
</table>

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 of annotated relations \( \text{ANN}(r) \) and \( \text{ANN}(r') \), denoted \( \text{ANN}(r) - \text{ANN}(r') \), is defined as \( \text{ANN}(r'') = \{ at \in \text{ANN}(r) \mid (\forall at' \in \text{ANN}(r')) (d \neq d' \lor t \neq t') \} \).

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

<table>
<thead>
<tr>
<th>Data</th>
<th>H</th>
<th>Day</th>
<th>Month</th>
<th>Year</th>
<th>( L_t )</th>
<th>( U_t )</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>1</td>
<td>8</td>
<td>1997</td>
<td>0.32</td>
<td>0.44</td>
<td></td>
</tr>
<tr>
<td>D1</td>
<td>5</td>
<td>8</td>
<td>1997</td>
<td>0.10</td>
<td>0.20</td>
<td></td>
</tr>
</tbody>
</table>

Suppose \( at_1 = (d, t, 0.2, 0.4) \in \text{ANN}(r) \) and \( at_2 = (d, t, 0, 1) \in \text{ANN}(r') \). Then by definition, \( \text{ANN}(r'') \) will not contain \( at_1 \). Now suppose we removed from \( \text{ANN}(r') \) all annotated tuples where \([L'_t, U'_t] = [0, 1] \). Here, \( \text{ANN}(r'') \) 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 \((e_1 \land e_2) \) at time \( t \), we must apply a probabilistic conjunction strategy \( \alpha \), i.e., \([L', U'] = [L_1, U_1] \oplus_\alpha [L_2, U_2] \) (§2.5). This allows users to ask queries such as “Compute the cartesian product of annotated relations \( \text{ANN}(r) \) and \( \text{ANN}(r') \) 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 relations \( \text{ANN}(r) \) and \( \text{ANN}(r') \) under the \( \alpha \) probabilistic conjunction strategy, denoted \( \text{ANN}(r) \times_\alpha \text{ANN}(r') \),
is defined as $\text{ANN}(r'') = \{(d'', t, L''_t, U''_t) \mid (\exists at \in \text{ANN}(r)) \land (\exists at' \in \text{ANN}(r')) \land (d'' = (\mathcal{P}(d), \mathcal{P}(d'), h'')) \land (h'' = (d.H \parallel d'.H)) \land ([L'_t, U'_t] = [L_t, U_t] \odot_a [L'_t, U'_t])\}$.  

**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 $\text{ANN}(r)$ (i.e., $\mathcal{P}(d)$), (ii) manifest data fields from $\text{ANN}(r')$ (i.e., $\mathcal{P}(d')$), and (iii) $h'' = d.H \parallel d'.H$ (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 $\alpha$. 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, $\text{ANN}(r_1) \times_{ig} \text{ANN}(r_2)$ will be

<table>
<thead>
<tr>
<th>$r_1$.Data</th>
<th>$r_2$.Data</th>
<th>H</th>
<th>Day</th>
<th>Month</th>
<th>Year</th>
<th>$l_d$</th>
<th>$u_d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>D1</td>
<td>2</td>
<td>8</td>
<td>1997</td>
<td>0.00</td>
<td>0.22</td>
<td></td>
</tr>
<tr>
<td>D1</td>
<td>D1</td>
<td>3</td>
<td>8</td>
<td>1997</td>
<td>0.00</td>
<td>0.11</td>
<td></td>
</tr>
<tr>
<td>D1</td>
<td>D1</td>
<td>6</td>
<td>8</td>
<td>1997</td>
<td>0.00</td>
<td>0.20</td>
<td></td>
</tr>
<tr>
<td>D1</td>
<td>D1</td>
<td>7</td>
<td>8</td>
<td>1997</td>
<td>0.00</td>
<td>0.20</td>
<td></td>
</tr>
<tr>
<td>D1</td>
<td>D1</td>
<td>8</td>
<td>8</td>
<td>1997</td>
<td>0.00</td>
<td>0.20</td>
<td></td>
</tr>
</tbody>
</table>

but $\text{ANN}(r_1) \times_{pc} \text{ANN}(r_2)$ will be

<table>
<thead>
<tr>
<th>$r_1$.Data</th>
<th>$r_2$.Data</th>
<th>H</th>
<th>Day</th>
<th>Month</th>
<th>Year</th>
<th>$l_d$</th>
<th>$u_d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>D1</td>
<td>2</td>
<td>8</td>
<td>1997</td>
<td>0.10</td>
<td>0.22</td>
<td></td>
</tr>
<tr>
<td>D1</td>
<td>D1</td>
<td>3</td>
<td>8</td>
<td>1997</td>
<td>0.05</td>
<td>0.11</td>
<td></td>
</tr>
<tr>
<td>D1</td>
<td>D1</td>
<td>6</td>
<td>8</td>
<td>1997</td>
<td>0.10</td>
<td>0.20</td>
<td></td>
</tr>
<tr>
<td>D1</td>
<td>D1</td>
<td>7</td>
<td>8</td>
<td>1997</td>
<td>0.10</td>
<td>0.20</td>
<td></td>
</tr>
<tr>
<td>D1</td>
<td>D1</td>
<td>8</td>
<td>8</td>
<td>1997</td>
<td>0.10</td>
<td>0.20</td>
<td></td>
</tr>
</tbody>
</table>

### 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 $\text{ANN}(r)$ 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. $\text{ANN}(r)$ and let “$A_1, \ldots, A_n$” be the (possibly empty) list of all manifest data fields which appear in the primary key of $\text{ANN}(r)$ but do not appear in $\mathcal{F}$. Then the **projection of field list $\mathcal{F}$ on annotated relation $\text{ANN}(r)$**, denoted $\pi_\mathcal{F}(\text{ANN}(r))$, is defined as $\text{ANN}(d'') = \{(d'', t, L''_t, U''_t) \mid (\exists at \in \text{ANN}(r)) \land (d'' = (\pi_\mathcal{F}(\mathcal{P}(d)), h'')) \land (h'' = (d.H \parallel [A_1:d.A_1, \ldots, A_n:d.A_n]))\}$.  

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 $\mathcal{F} = \{\text{Data1}\}$ and if our primary key for $\text{ANN}(r_3)$ was “Data1,Data2”, then $\text{ANN}(r'') = \pi_\mathcal{F}(\text{ANN}(r_3))$ will be
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 a 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 since our framework only allows instantaneous events (§2.4). However, our algebra is rich enough to express it indirectly by (i) setting each $h''$ in $\text{ANN}(r''')$ to EMPTY and then (ii) invoking a disjunctive p-strategy based compaction (§5.1) on the result of step (i). The resulting annotated relation may be inconsistent. All tuples in the result of this operation denote instantaneous events.

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 $C$ be defined as $((\text{ANN}(r),L_1 = \text{ANN}(r').L_1) \wedge \ldots \wedge (\text{ANN}(r),L_n = \text{ANN}(r').L_n))$ where “$L_1 \ldots L_n$” is the list of all manifest data fields which occur in the schema for both $\text{ANN}(r)$ and $\text{ANN}(r')$. Then the join of annotated relations $\text{ANN}(r)$ and $\text{ANN}(r')$ under the $\alpha$ probabilistic conjunction strategy, denoted $\pi_{\mathcal{F}}(\sigma_C(\text{ANN}(r) \times_\alpha \text{ANN}(r')))$, is defined as $\pi_{\mathcal{F}}(\sigma_C(\text{ANN}(r) \times_\alpha \text{ANN}(r')))$. For example, $\text{ANN}(r''') = \text{ANN}(r_3) \Rightarrow_{p,c} \text{ANN}(r_4)$ will be

<table>
<thead>
<tr>
<th>Data1</th>
<th>Data2</th>
<th>H</th>
<th>Day</th>
<th>Month</th>
<th>Year</th>
<th>$L_t$</th>
<th>$U_t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>D2</td>
<td>2</td>
<td>8</td>
<td>1997</td>
<td>0.10</td>
<td>0.40</td>
<td></td>
</tr>
<tr>
<td>D1</td>
<td>D3</td>
<td>3</td>
<td>8</td>
<td>1997</td>
<td>0.15</td>
<td>0.20</td>
<td></td>
</tr>
</tbody>
</table>

Notice that all of the hidden fields in $\text{ANN}(r''')$ above are EMPTY. This occurs since in our example, $\mathcal{F} = \{\text{Data1},\text{Data2}\}$ 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 $C$ and $\mathcal{F}$.

### 6 TP-Algebra

This is the most important section of the paper. As we have 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[13], 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, join, and tightening of TP-relations. With the exception of TP-compression and tightening (which have no analogs 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. The correctness theorems are stated in this section and the proofs are given in Appendix C.

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

Furthermore, binary TPA operator $op^T$ correctly implements the semantics for binary TATA operator $op^A$ iff $\text{ANN}(r op^T r') = \text{ANN}(r) op^A \text{ANN}(r')$ 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 $C$ is not an inapplicable condition, and projections are only defined when field list $F$ 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 $\gamma$ contain $n$ TP-cases of the form $\gamma_i = \langle C_i; D_i; L_i; U_i; e_i \rangle \in \gamma$ and let $\gamma'$ contain $n'$ TP-cases of the form $\gamma'_j = \langle C'_j; D'_j; L'_j; U'_j, e'_j \rangle \in \gamma'$.

### 6.1 TP-compression of a TP-relation

The basic idea behind TP-compression is to allow the data in a TP-relation (either base or derived) to be compressed. Let $N(r)$ and $N(tp)$ denote the number of TP-cases in TP-relation $r$ and TP-tuple $tp$ respectively. When we apply a TP-compression function to $r$, we may be able to reduce the size of $N(r)$.

**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 $\text{ANN}(r)$ and $\text{ANN}(r'')$ which maps each $(d, t, L_i, U_i) \in \text{ANN}(r)$ to a $(d, t, L_i, U_i'') \in \text{ANN}(r'')$ such that $L_i \leq U_i'' \leq U_i$.

In other words, $\text{ANN}(r'')$ and $\text{ANN}(r)$ 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 function combines TP-cases which share the same distribution.

**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
Theorem 6

**Deﬁnition 6.4 (TP-compression of a TP-relation; hybrid)** The hybrid TP-compression of TP-relation \( r \), denoted \( \Xi^{hy}(r) \), is equal to the multiset \( S \) which can be constructed in the following way: Initially, let \( S = r'' \). Then for each \( (d, \gamma'') \in r'' \) and for each pair of TP-cases \( \gamma_i = (C_i, D_i, L_i, U_i, \delta_i), \gamma_j = (C_j, D_j, L_j, U_j, \delta_j) \in \gamma'' \) where \( \text{sol}(D_i) = \text{sol}(D_j) \), remove \( \gamma_i, \gamma_j \) from \( \gamma'' \) and add TP-case \( \langle (C_i \land C_j), D_i, L_i, U_i, \delta \rangle \) to \( \gamma'' \).

Another possible TP-compression function takes advantage of the uniform distribution’s regularity.

**Deﬁnition 6.5 (TP-compression of a TP-relation; hybrid)** The hybrid TP-compression of TP-relation \( r \), denoted \( \Xi^{hy}(r) \), is equal to the multiset \( S \) which can be constructed in the following way: Initially, let \( S = r'' \). Then for each \( (d, \gamma'') \in r'' \) and for each pair of TP-cases \( \gamma_i = (C_i, D_i, L_i, U_i, \delta_i), \gamma_j = (C_j, D_j, L_j, U_j, \delta_j) \in \gamma'' \) where \( n_i = |\text{sol}(D_i)|, n_j = |\text{sol}(D_j)| \) and \( \left[ L_i, U_i \right] = \frac{1}{n_i} \left[ L_i, U_i \right], \left[ L_j, U_j \right] \), remove \( \gamma_i, \gamma_j \) from \( \gamma'' \) and add TP-case \( \langle (C_i \land C_j), (D_i \lor D_j), L_{i,j}, \min(1, U_{i,j}), \delta \rangle \) to \( \gamma'' \) where \( n_{ij} = \left[ \text{sol}(D_i \lor D_j) \right] \) and \( \left[ L_{i,j}, U_{i,j} \right] = n_{ij} \left[ L_i, U_i \right] \).

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

**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 \( S \), where for each \( 1 \leq i < n, t_{i+1} = n x t_r(t_i) \). Then if \( \gamma'' \) contains \( n \) TP-cases of the form \( \langle (\#), (t_i), L''_{t_i}, U''_{t_i}, \delta \rangle \) where \( \left[ L''_{t_i}, U''_{t_i} \right] = p \cdot \left[ L_{t_i}, U_{t_i} \right] \) for all \( 1 \leq i < n \) and \( (U_{t_i} \div p \leq 1) \), apply an operator which replaces these \( n \) TP-cases with the TP-case \( \langle (\#), (t_1 \sim t_n), L''_{t_i}, U''_{t_i}, \delta'' \rangle \) where \( \left[ L'', U'' \right] = \left[ L_{t_i} \div p, U_{t_i} \div p \right] \) and \( \delta'' = "g", p" \).

The aforementioned operator performs a \( g \)-based TP-compression. Note that for any distribution function \( \delta \), one can deﬁne a corresponding \( \delta \)-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 \( \kappa \) 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 \( \text{ANN}(\kappa(r)) = \text{ANN}(r) \).
- **Conservativeness**: If \( at = (d, t, L_i, U_i) \in \text{ANN}(\kappa(r)) \), then \( \exists at' = (d, t, L'_i, U'_i) \in \text{ANN}(r) \).
The Compactness, NFA, and Conservativeness axioms for TP-relations are similar in spirit and intuition to the same concepts defined earlier for annotated relations.

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

**Definition 6.7 (\(\chi\)-compaction of a TP-relation)** Let $\chi$ be a combination function. Then the $\chi$-compaction of TP-relation $r$, denoted $\kappa_\chi(r)$, is defined as $\text{ANN}(\kappa_\chi(r)) = \{at = (d, t, L_t, U_t) \mid [L_t, U_t] = \chi([[L_1^{(d,t)}, U_1^{(d,t)}], \ldots, [L_k^{(d,t)}, U_k^{(d,t)}]])\}$ where $\text{ANN}(r)[d, t] = \{at_1^{(d,t)}, \ldots, at_k^{(d,t)}\}$ and $at_i^{(d,t)} = (d, t, i^{(d,t)}, U_i^{(d,t)})$.

The following lemma states that an operation defined in this manner is indeed a compaction operation.

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

Note that some parts of Definitions 6.6 and 6.7 were presented in a declarative manner — via the contents of the annotation of the result. 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 $\chi$ or a p-strategy $\rho$. The boxed line in this algorithm shows exactly where a combination function or p-strategy is applied to compact data-identical tuples. Note that when $f$ is a p-strategy, the application of $f$ to a set $X$ of intervals merely represents the iterative application of $f$ to pairs of intervals in $X$ — as p-strategies are associative and commutative, this is well defined.

The following states that this algorithm correctly computes a $\kappa_\chi(r)$ compaction.

**Theorem 7** Let $\chi$ be a combination function. Then algorithm **Compute-Compaction**\((r, \chi)\) 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 $[L_t, U_t] = ([L_1^{(d,t)}, U_1^{(d,t)}] \circ \ldots \circ [L_k^{(d,t)}, U_k^{(d,t)}])$ and let $[L_t, U_t] = ([L_1^{(d,t)}, U_1^{(d,t)}] \circ \ldots \circ [L_k^{(d,t)}, U_k^{(d,t)}])$ when defining $\kappa_\circ(r)$ and $\kappa_\oplus(r)$ respectively.

**Lemma 2** Let $\rho$ 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 $\rho$ be a (conjunctive or disjunctive) p-strategy. Then algorithm **Compute-Compaction**\((r, \rho)\) 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 $(\kappa^A(\text{ANN}(r)), \kappa^T(r))$ of compaction operators is a **compatible pair** iff for every TP-relation $r$, $\kappa^A(\text{ANN}(r)) = \text{ANN}(\kappa^T(r))$. \hfill $\Box$
In this section, we show how we can correctly implement the intersection of two TP-relations. Intersection consists 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 $\Gamma = \Gamma' = \emptyset$.

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

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

**Theorem 10** Let $\rho$ be any $p$-strategy. Then $\langle \rho(\text{ANN}(r)), \rho(r) \rangle$ is a compatible pair.
2. For each \( \gamma_i \in \gamma \) and each \( \gamma'_j \in \gamma' \) where \(|\text{sol}(C_i \land C'_j)| \geq 1\), add TP-case \(((C_i \land C'_j), D_i, L_i, U_i, \delta_i)\) to \( \Gamma \) and add TP-case \(((C_i \land C'_j), D'_j, L'_j, U'_j, \delta'_j)\) to \( \Gamma' \). Note that \((C_i \land C'_j)\) is shared by both TP-cases.

3. If \( \Gamma \neq \emptyset \), add TP-tuples \((d, \Gamma)\) and \((d', \Gamma')\) to \( r'' \). Note that \( \Gamma \neq \emptyset \Rightarrow \Gamma' \neq \emptyset \). \( \Gamma \) will be empty if there are no overlapping time points.

For example, \( r'' = r_1 \cap r_2 \) will be

<table>
<thead>
<tr>
<th>Data</th>
<th>H</th>
<th>C</th>
<th>D</th>
<th>L</th>
<th>U</th>
<th>( \delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>(#)</td>
<td>(2/8/1997 ~ 3/8/1997)</td>
<td>(1/8/1997 ~ 3/8/1997)</td>
<td>0.64</td>
<td>0.88</td>
<td>g</td>
</tr>
<tr>
<td></td>
<td>(#)</td>
<td>(6/8/1997 ~ 8/8/1997)</td>
<td>(5/8/1997 ~ 8/8/1997)</td>
<td>0.40</td>
<td>0.80</td>
<td>u</td>
</tr>
</tbody>
</table>

As in the case of the TATA, we apply a \( \kappa_\chi \) compaction operator to the result of a multiset intersection.

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

In order to keep the size of \( r \cap \chi r' \) manageable, we usually perform a TP-compression on the result of a compaction.

For example, \( \Xi^{\kappa_\chi} (r \cap \chi r') = \Xi^{\kappa_\chi} (\kappa_\chi (r'')) \) will be

<table>
<thead>
<tr>
<th>Data</th>
<th>H</th>
<th>C</th>
<th>D</th>
<th>L</th>
<th>U</th>
<th>( \delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>(#)</td>
<td>(2/8/1997 ~ 2/8/1997)</td>
<td>(2/8/1997 ~ 3/8/1997)</td>
<td>0.16</td>
<td>0.22</td>
<td>u</td>
</tr>
<tr>
<td></td>
<td>(#)</td>
<td>(3/8/1997 ~ 6/8/1997)</td>
<td>(3/8/1997 ~ 8/8/1997)</td>
<td>0.08</td>
<td>0.11</td>
<td>u</td>
</tr>
<tr>
<td></td>
<td>(#)</td>
<td>(6/8/1997 ~ 8/8/1997)</td>
<td>(6/8/1997 ~ 9/8/1997)</td>
<td>0.30</td>
<td>0.60</td>
<td>u</td>
</tr>
</tbody>
</table>

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)** \( \text{ANN}(r \cap \chi r') = \text{ANN}(r) \cap \chi \text{ANN}(r') \).

### 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 \( r'' = r \oplus r' \).

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

<table>
<thead>
<tr>
<th>Data</th>
<th>H</th>
<th>C</th>
<th>D</th>
<th>L</th>
<th>U</th>
<th>( \delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>(#)</td>
<td>(1/8/1997 ~ 3/8/1997)</td>
<td>(1/8/1997 ~ 3/8/1997)</td>
<td>0.64</td>
<td>0.88</td>
<td>g</td>
</tr>
<tr>
<td></td>
<td>(#)</td>
<td>(5/8/1997 ~ 8/8/1997)</td>
<td>(5/8/1997 ~ 8/8/1997)</td>
<td>0.40</td>
<td>0.80</td>
<td>u</td>
</tr>
<tr>
<td>D1</td>
<td>(#)</td>
<td>(2/8/1997 ~ 3/8/1997)</td>
<td>(2/8/1997 ~ 3/8/1997)</td>
<td>0.20</td>
<td>0.50</td>
<td>g</td>
</tr>
</tbody>
</table>
As in the case of the TATA, we apply a $\kappa_\chi$ 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 $\chi$ combination function, denoted $r \cup_\chi r'$, is defined as $\kappa_\chi(r \cup r')$.

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

<table>
<thead>
<tr>
<th>Data</th>
<th>$H$</th>
<th>$C$</th>
<th>$L$</th>
<th>$U$</th>
<th>$t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td></td>
<td>(1/8/1997)</td>
<td>(1/8/1997 ~ 3/8/1997)</td>
<td>0.64</td>
<td>0.88</td>
</tr>
<tr>
<td></td>
<td>(♯)</td>
<td>(2/8/1997 ~ 2/8/1997)</td>
<td>0.16</td>
<td>0.22</td>
<td>$u$</td>
</tr>
<tr>
<td></td>
<td>(♯)</td>
<td>(3/8/1997 ~ 3/8/1997)</td>
<td>0.08</td>
<td>0.11</td>
<td>$u$</td>
</tr>
<tr>
<td></td>
<td>(♯)</td>
<td>(5/8/1997 ~ 9/8/1997)</td>
<td>0.50</td>
<td>1.00</td>
<td>$u$</td>
</tr>
</tbody>
</table>

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

**Theorem 12 (Correctness of union)** $\text{ANN}(r) \cup_\chi \text{ANN}(r') = \text{ANN}(r \cup_\chi r')$.

### 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 $\gamma_i = \langle C_i, D_i, L_i, U_i, \hat{c}_i \rangle$ be a TP-case, let $\bar{C} = (F \ op \ v)$ 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 $\gamma_i$ and $\bar{C}$ as input, and returns as output a temporal constraint $C_i''$ where

1. $\text{sol}(C_i'') \subseteq \text{sol}(C_i)$
2. For each time point $t \in \text{sol}(C_i)$, $(x_t \ op \ v)$ must be true when $x_t = \hat{c}_i(D_i,t) \cdot x$
3. There is no temporal constraint $C_i'$ where $(\text{sol}(C_i) \supset \text{sol}(C_i''))$ and $C_i'$ satisfies the previous cases.

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

For example if $\gamma_i = \langle (♯), (5/8/1997 ~ 8/8/1997), 0.4, 0.8, q \rangle$ and $\bar{C} = (U > 0.15)$, TP-filter($\gamma_i$, $\bar{C}$) will be $C_i'' = (5/8/1997 ~ 6/8/1997)$ since $(0.4 > 0.15)$ for 5/8/1997 and $(0.2 > 0.15)$ for 6/8/1997.

In general, $n = |\text{sol}(C_i)|$ 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 $\hat{c}_i = “ag”$, then we only need to test one time point $t \in \text{sol}(C_i)$; if $t$ should be in sol($C_i''$), then $C_i'' = C_i$ or $C_i'' = \emptyset$ otherwise. This “all or none” behavior occurs since each $t \in \text{sol}(C_i)$ will have the same probability value after distributing uniformly.

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

We are now ready to define selection using atomic selection conditions.
Definition 6.14 (Selection on a TP-tuple; atomic condition) The selection of atomic condition $C$ on TP-tuple $t_p = (d, \gamma)$, denoted $\sigma_C(t_p)$, can be constructed in the following way: Initially, let $\gamma'' = \emptyset$.

- If $C$ is a data condition, let $\gamma'' = \gamma$ if $d$ satisfies $C$.
- If $C$ is a temporal condition, then for each $\gamma_i \in \gamma$ where $C''_{i, t} = (C_i \land C)$ is consistent, add TP-case $\langle C''_{i, t}, D_i, L_i, U_i, \delta_i \rangle$ to $\gamma''$.
- If $C$ is a probabilistic condition, then for each $\gamma_i \in \gamma$ where $C''_{i, t} = \text{TP-filter} (\gamma_i, C)$ is consistent, add TP-case $\langle C''_{i, t}, D_i, L_i, U_i, \delta_i \rangle$ to $\gamma''$.

If $\gamma'' = \emptyset$ then $\sigma_C(t_p) = \emptyset$. Otherwise, $\sigma_C(t_p) = (d, \gamma'')$. □

Definition 6.15 (Selection on a TP-relation; atomic condition) The selection of atomic condition $C$ on TP-relation $r = \{t_{p_1}, \ldots, t_{p_n}\}$, denoted $\sigma_C(r)$, is defined as $\langle \sigma_C(t_{p_1}) \uplus \cdots \uplus \sigma_C(t_{p_n}) \rangle$.

Note that for all $t_{p_i}, t_{p_j} \in r$, $\sigma_C(t_{p_i})$ does not affect the results of $\sigma_C(t_{p_j})$ when computing $\sigma_C(r)$.

For example if $C = (2/8/1997 \sim 7/8/1997)$, $\sigma_C(r_1)$ will be

<table>
<thead>
<tr>
<th>Data</th>
<th>H</th>
<th>C</th>
<th>D</th>
<th>L</th>
<th>U</th>
<th>δ</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td></td>
<td>(2/8/1997 ~ 3/8/1997)</td>
<td>(1/8/1997 ~ 3/8/1997)</td>
<td>0.64</td>
<td>0.88</td>
<td>g</td>
</tr>
</tbody>
</table>

and $\sigma_C(r_2)$ will be

<table>
<thead>
<tr>
<th>Data</th>
<th>H</th>
<th>C</th>
<th>D</th>
<th>L</th>
<th>U</th>
<th>δ</th>
</tr>
</thead>
</table>

but if $C = (L \neq 0.10)$, $\sigma_C(r_1)$ will be

<table>
<thead>
<tr>
<th>Data</th>
<th>H</th>
<th>C</th>
<th>D</th>
<th>L</th>
<th>U</th>
<th>δ</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td></td>
<td>(#)</td>
<td>(1/8/1997 ~ 3/8/1997)</td>
<td>0.64</td>
<td>0.88</td>
<td>g</td>
</tr>
</tbody>
</table>

and $\sigma_C(r_2)$ will be

<table>
<thead>
<tr>
<th>Data</th>
<th>H</th>
<th>C</th>
<th>D</th>
<th>L</th>
<th>U</th>
<th>δ</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td></td>
<td>(3/8/1997)</td>
<td>(2/8/1997 ~ 3/8/1997)</td>
<td>0.20</td>
<td>0.50</td>
<td>g</td>
</tr>
</tbody>
</table>

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 $C$ on TP-relation $r$, denoted $\sigma_C(r)$, is defined inductively in the following way:

- If $C$ is an atomic condition, then $\sigma'' = \sigma_C(r)$ by way of our previous definition.
- If $C$ is of the form $(C_1 \land C_2)$, then $\sigma'' = \sigma_{C_1}(\sigma_{C_2}(r))$. 

37
If \( C \) is of the form \((C_1 \lor C_2)\), then \( r'' = \sigma_{C_1}(r) \cup_{eq} \sigma_{C_2}(r)\). (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 with any other combination without the result being changed.)

If \( C \) is of the form \((\neg C_1)\), then

- If \( C_1 \) is of the form \((C_2 \land C_3)\), \((C_2 \lor C_3)\), or \((\neg C_2)\), then \( r'' = \sigma_{C_4}(r)\) where \( C_4 = \neg C_2 \lor \neg C_3, C_4 = \neg C_2 \land \neg C_3, \) or \( C_4 = (\neg C_2) \) respectively.
- If \( C_1 \) is a data, temporal, or probabilistic condition, then \( r'' = \sigma_{C_4}(r)\) where \( C_4 \) is the atomic, logical negation of \( C_1 \).
- Otherwise, \( C_1 \) is an inapplicable condition and so \( r'' \) is not defined.

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

For example if \( C_1 = (2/8/1997 \sim 7/8/1997), C_2 = (L \neq 0.10) \), and \( C = (C_1 \land C_2) \), then \( \sigma_C(r_1) \) will be

<table>
<thead>
<tr>
<th>Data</th>
<th>H</th>
<th>C</th>
<th>D</th>
<th>L</th>
<th>U</th>
<th>?</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td></td>
<td>(2/8/1997 \sim 3/8/1997)</td>
<td>(1/8/1997 \sim 3/8/1997)</td>
<td>0.64</td>
<td>0.88</td>
<td>g</td>
</tr>
</tbody>
</table>

and \( \sigma_C(r_2) \) will be

<table>
<thead>
<tr>
<th>Data</th>
<th>H</th>
<th>C</th>
<th>D</th>
<th>L</th>
<th>U</th>
<th>?</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td></td>
<td>(3/8/1997)</td>
<td>(2/8/1997 \sim 3/8/1997)</td>
<td>0.20</td>
<td>0.50</td>
<td>g</td>
</tr>
</tbody>
</table>

but if \( C = (C_1 \lor C_2) \), \( \Xi^{by}(\sigma_C(r_1)) \) will be

<table>
<thead>
<tr>
<th>Data</th>
<th>H</th>
<th>C</th>
<th>D</th>
<th>L</th>
<th>U</th>
<th>?</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td></td>
<td>(1/8/1997 \sim 3/8/1997)</td>
<td>(1/8/1997 \sim 3/8/1997)</td>
<td>0.64</td>
<td>0.88</td>
<td>g</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(5/8/1997 \sim 7/8/1997)</td>
<td>(5/8/1997 \sim 8/8/1997)</td>
<td>0.40</td>
<td>0.80</td>
<td>u</td>
</tr>
</tbody>
</table>

and \( \Xi^{by}(\sigma_C(r_2)) \) will be

<table>
<thead>
<tr>
<th>Data</th>
<th>H</th>
<th>C</th>
<th>D</th>
<th>L</th>
<th>U</th>
<th>?</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td></td>
<td>(2/8/1997 \sim 3/8/1997)</td>
<td>(2/8/1997 \sim 3/8/1997)</td>
<td>0.20</td>
<td>0.50</td>
<td>g</td>
</tr>
</tbody>
</table>

The following theorem states that our definition of selection preserves commutativity.

**Theorem 13** \( \sigma_{C_1}(\sigma_{C_2}(r)) = \sigma_{C_2}(\sigma_{C_1}(r)) \).

The following table shows how one may generate queries on 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 assume that \( r \) uses two TP-tuples for each event \( e \); one for \( \text{st}(e) \) (where the value of the “Kind” field is “S”), and one for \( \text{end}(e) \) (where the value of the “Kind” field is “E”). Assume that for for each event \( e \), if \( \text{st}(e) \) or \( \text{end}(e) \) satisfies the selection condition, then the TP-tuples for both \( \text{st}(e) \) and \( \text{end}(e) \) should be included in the result. Our queries will return every event \( e \) which satisfies some relationship w.r.t. event \( e_q \) where \( \text{st}(e_q) = t_1 \) and \( \text{end}(e_q) = t_2 \).
The following shows that our definition of selection correctly implements the TATA semantics.

**Theorem 14 (Correctness of selection)** $\text{ANN}(\sigma_C(r)) = \sigma_C(\text{ANN}(r))$.

### 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 $C' = (C'_1 \lor \ldots \lor C'_{n'})$. Recall that $tp'$ contains exactly $n'$ TP-cases.
2. For each $\gamma_i \in \gamma$ where $C''_i = (C_i \land \neg C')$ is consistent, add TP-case $\langle C''_i, D_i, L_i, U_i, b_i \rangle$ to $\gamma''$.
3. Remove $tp$ from $r''$. Then if $\gamma'' \neq \emptyset$, add TP-tuple $(d, \gamma'')$ to $r''$.

For example $r_1 - r_2$ will be

<table>
<thead>
<tr>
<th>Data</th>
<th>H</th>
<th>C</th>
<th>D</th>
<th>L</th>
<th>U</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>(1/8/1997)</td>
<td>1/8/1997 \sim 3/8/1997</td>
<td>0.64</td>
<td>0.88</td>
<td>g</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(5/8/1997)</td>
<td>5/8/1997 \sim 8/8/1997</td>
<td>0.40</td>
<td>0.80</td>
<td>u</td>
<td></td>
</tr>
</tbody>
</table>

and $r_2 - r_1$ will be

<table>
<thead>
<tr>
<th>Data</th>
<th>H</th>
<th>C</th>
<th>D</th>
<th>L</th>
<th>U</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>(9/8/1997)</td>
<td>6/8/1997 \sim 9/8/1997</td>
<td>0.40</td>
<td>0.80</td>
<td>u</td>
<td></td>
</tr>
</tbody>
</table>

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

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

39
6.7 Cartesian product of two TP-relations

In this section, we show how we can correctly implement the cartesian product of two TP-relations. Recall, as in the annotated case, that when taking the cartesian product of two relations, we must know the relationship, if any, between the events denoted by the tuples in the two relations because the probability of a concatenated tuple being present in the result of the Cartesian Product is the probability of a conjunctive event. Thus, conjunction strategies parametrize the Cartesian Product operation.

Definition 6.18 (cartesian product of two TP-relations) The cartesian product of TP-relations \( r \) and \( r' \) under the \( \alpha \) 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 \( [L_i, U_i] = [L_i \cdot x_i, U_i \cdot x_i] \) where \( x_i = \delta_i(D_i, t) \).
   (b) Let \( [L_j', U_j'] = [L_j' \cdot x_j', U_j' \cdot x_j'] \) where \( x_j' = \delta_j'(D_j', t) \).
   (c) Let \( [L''_i, U''_i] = ([L_i, U_i] \times_\alpha [L'_j, U'_j]) \).
   (d) Add TP-case \( ((\#), (t), L''_i, U''_i, \gamma'') \) to \( \gamma'' \).
3. If \( \gamma'' \neq \emptyset \), add TP-tuple \( (d'', \gamma'') \) to \( \gamma'' \) where \( d'' = (P(d), P(d'), h'') \) and \( h'' = (d, H \parallel d', H) \).

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

<table>
<thead>
<tr>
<th>( r_1 ).Data</th>
<th>( r_2 ).Data</th>
<th>H</th>
<th>C</th>
<th>D</th>
<th>L</th>
<th>U</th>
<th>( \delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>D1</td>
<td>(#)</td>
<td>(2/8/1997 ~ 2/8/1997)</td>
<td>0.00</td>
<td>0.22</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(#)</td>
<td>(3/8/1997 ~ 3/8/1997)</td>
<td>0.00</td>
<td>0.11</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(#)</td>
<td>(6/8/1997 ~ 8/8/1997)</td>
<td>0.00</td>
<td>0.60</td>
<td>n</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>( r_1 ).Data</th>
<th>( r_2 ).Data</th>
<th>H</th>
<th>C</th>
<th>D</th>
<th>L</th>
<th>U</th>
<th>( \delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>D1</td>
<td>(#)</td>
<td>(2/8/1997 ~ 2/8/1997)</td>
<td>0.10</td>
<td>0.22</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(#)</td>
<td>(3/8/1997 ~ 3/8/1997)</td>
<td>0.05</td>
<td>0.11</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(#)</td>
<td>(6/8/1997 ~ 8/8/1997)</td>
<td>0.30</td>
<td>0.60</td>
<td>n</td>
<td></td>
</tr>
</tbody>
</table>

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 \( F \) 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 \( F \). Then the projection of field list \( F \) on TP-relation \( r \), denoted \( / F / (r) \), can be constructed in the following way: Initially, let \( r / 0/= / \). Then for each \( (d, \gamma) \in r \), add TP-tuple \( (d / 0/= /; \gamma) \) to \( r / 0/= / \) where \( d / 0/= / = (\pi_F(\mathcal{P}(d)), h^\gamma) \) and \( h^\gamma = (d.\mathcal{H} \mid “A_1:d.A_1, \ldots, A_n:d.A_n”). \)

Recall that the \( \pi_F(\mathcal{P}(d)) \) operator was defined in section 5.7.

For example if \( F = “\text{Data1}” \) and our primary key for \( r_3 \) was “\( \text{Data1,Data2} \)”, \( r_3 = \pi_F(r_3) \) will be

<table>
<thead>
<tr>
<th>Data1</th>
<th>Data2:Data2</th>
<th>(#)</th>
<th>(2/8/1997 ~ 2/8/1997)</th>
<th>0.20</th>
<th>0.40</th>
<th>u</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>Data2:Data3</td>
<td>(#)</td>
<td>(2/8/1997 ~ 3/8/1997)</td>
<td>0.60</td>
<td>0.80</td>
<td>g</td>
</tr>
</tbody>
</table>

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

Theorem 17 (Correctness of projection) \( ANN(\pi_F(r)) = \pi_F(ANN(r)) \).

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 \( C \) be \( ((r.\mathcal{L}_1 = r’.\mathcal{L}_1) \land \ldots \land (r.\mathcal{L}_n = r’.\mathcal{L}_n)) \) 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 \( \alpha \) probabilistic conjunction strategy, denoted \( r \bowtie_{\alpha} r’ \), is defined as \( \pi_F(\sigma^\alpha_C(r \times_o r’)) \) where \( F \) 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

<table>
<thead>
<tr>
<th>Data1</th>
<th>Data2</th>
<th>H</th>
<th>C</th>
<th>D</th>
<th>L</th>
<th>U</th>
<th>( \gamma )</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>D2</td>
<td>(#)</td>
<td>(2/8/1997)</td>
<td>0.10</td>
<td>0.40</td>
<td>u</td>
<td></td>
</tr>
<tr>
<td>D1</td>
<td>D3</td>
<td>(#)</td>
<td>(3/8/1997)</td>
<td>0.15</td>
<td>0.20</td>
<td>u</td>
<td></td>
</tr>
</tbody>
</table>

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

Theorem 18 (Correctness of join) \( r \bowtie_{\alpha} r’ = ANN(r) \bowtie_{\alpha} ANN(r’) \).

6.10 Tightening of a TP-relation

Recall that TP-tuples can be “loose” in the sense that a TP-tuple may, for example, have two TP-cases, \( \gamma_1, \gamma_2 \) with \( [L_1, U_1] = [0.3, 0.6] \) and \( [L_2, U_2] = [0.5, 0.8] \). In this case, it is easy to see that we can tighten the upper bounds of these ranges, and reset them to \( [0.3, 0.5] \) and \( [0.5, 0.7] \) respectively. The reason is that the
two lower bounds sum up to 0.8, thus allowing us an upper bound in each case that is no more than 0.2 more than the lower bound. This is a simple example of tightening. Recall that tightening played in establishing Theorem 4, which showed that the annotated expansion of a TP-relation faithfully represents the TP-relation as long as the TP-relation is compact and tight.

In this section, we describe a procedure for tightening a compact TP-relation $r$. Tight TP-relation $r''$ can be constructed from $r$ in the following way: Initially, let $r'' = \emptyset$ and let $r = \kappa_\chi(r)$ for some combination function $\chi$. Then for each $tp \in r$, add Tighten-TP-Tuple$(tp)$ to $r''$. An algorithm for Tighten-TP-Tuple is presented below.

Note that for each $tp_1, tp_2 \in r$, Tighten-TP-Tuple$(tp_1)$ and Tighten-TP-Tuple$(tp_2)$ do not affect each other since $r$ is compact and hence $tp_1$ and $tp_2$ must refer to different events. Also note that Tighten-TP-Tuple works much faster when the distribution functions are uniform since here, all upper bounds for a single TP-case will be tightened by the same amount.

```plaintext
Algorithm Tighten-TP-Tuple(tp):
Input: TP-tuple $tp = (d, \gamma)$ where $\gamma = \{\gamma_1, \ldots, \gamma_n\}$ and for all $1 \leq i \leq n, \gamma_i = \{C_i, D_i, L_i, U_i, \delta_i\}$
Output: Tight TP-tuple $tp''$ which is a tightening of $tp$

1. $L := 0; U := 0; \text{// } [L, U]$ will hold the sum of the lower and upper bounds
2. for $i := 1$ to $n$ do
3. \hspace{.5cm} $L'_i := \delta_i(D_i, C_i) \cdot L_i; \ \ L := L + L'_i$
4. \hspace{.5cm} $U'_i := \delta_i(D_i, C_i) \cdot U_i; \ \ U := U + U'_i$
5. if $U < 1.0$ then return $tp'' := tp; \text{// If } U < 1.0, \text{ then } tp$ was already tight
6. $\gamma'' = \emptyset$
7. for $i := 1$ to $n$ do
8. \hspace{.5cm} if $\delta_i \neq u$ then
9. \hspace{.5cm} \hspace{.5cm} foreach $t \in \text{sol}(C_i)$ { 
10. \hspace{.5cm} \hspace{1cm} $L_i := \delta_i(D_i, t) \cdot L_i; \ \ U_i := \delta_i(D_i, t) \cdot U_i; \ \ U' := 1 - (L - L_i);
11. \hspace{.5cm} \hspace{1cm} if $U' < U_i$ then $U_i := U'$
12. \hspace{.5cm} \hspace{1cm} Add TP-case $\langle \#, (i), L_i, U_i, u \rangle$ to $\gamma''$; 
13. \hspace{.5cm} \hspace{.5cm} else 
14. \hspace{.5cm} \hspace{1cm} $m = |\text{sol}(D_i)|; \ \ L_i = \frac{L_i}{m}; \ \ U_i = \frac{U_i}{m};$
15. \hspace{.5cm} \hspace{1cm} $U' := 1 - (L - L_i); \ \ U'' := m \cdot \min(U_i, U');$
16. \hspace{.5cm} \hspace{1cm} Add TP-case $\langle \#, (i), L_i, U''_i, \delta_i \rangle$ to $\gamma''$; 
17. \hspace{.5cm} \hspace{1cm} return $tp'' := (d, \gamma'')$
End-Algorithm
```

7 Implementation and Experiments

All of the 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. A sample screen dump is shown in Figure 4.

In the implementation, a TP-database is a collection of TP-relations which have the same chronon. Each TP-relation \( r \) is actually stored as two tables; the data-table \( r_d \) stores \( r \)'s data-items and hidden fields while the case-table \( r_c \) stores \( r \)'s TP-cases. Both of these tables have an indexed “Id” field which stores an integer used to reconstruct \( r \) from the join of \( r_d \) and \( r_c \). Basically, if two tuples have the same “Id”, then we assume that both tuples refer to the same event. Note however that unless we just performed a compaction, two tuples which refer to different events may have different “Id”s.

Temporal constraints are represented by an array of integers. Basically, each temporal constraint \( C \) is broken up into a set of \( n \) disjoint, non-adjacent ranges of time points. Each range consists of two time points (the starting and ending time) and each time point consists of two integers (the first represents the number of milliseconds since the start of a day while the second represents the number of days since some fixed reference date). Thus, the first element of \( C \)’s array will hold \( n \), the next four elements will hold \( C \)’s first range, the next four elements will hold \( C \)’s second range, etc. Here, ranges are stored in chronological order. Note that temporal constraints can be indexed by using an auxiliary data structure, e.g. a segment tree [40] or constraint indexing methods such as those in [5, 6, 22].

7.1 Experiments

We conducted two sets of experiments. The first set of experiments was intended to demonstrate 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 with \( C = D =, |\text{sol}(D)| == 1 \). and \( \delta = “u” \).

We should mention that 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 [10]), 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 [10]. In some of the charts shown in Appendix B 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 \( nT \)uples TP-tuples where \( nT \)uples \( \in \{100, 500, 1000\} \). Each TP-tuple had one TP-case \((C_i, D_i, L_i, U_i, b_i)\) where \( C_i = D_i = (t_1, ~ t_2) \), \( t_1 = \text{random}(\{t \in \text{sol}(1/1/1998 ~ 31/12/1998)\}) \), \( t_2 \) is the time point which occurs \( n\text{TimePoints} \) 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:

1. Intersection and Union Computations
   \( \Xi \cup (r \cap eq \ r') \), \( \text{ANN}(r) \cap eq \text{ANN}(r') \), \( \Xi \cup (r \cup eq \ r') \), and \( \text{ANN}(r) \cup eq \text{ANN}(r') \).

Chart (a) in Appendix B shows that intersection takes time that is more or less linear in the number of
Query mode
Database = Paper1, Chronon = day, Table = rel3

TP-tuples

<table>
<thead>
<tr>
<th>Data1/s</th>
<th>Data2/s</th>
<th>H/b</th>
<th>C/b</th>
<th>D/b</th>
<th>L/f</th>
<th>U/f</th>
<th>Delta/s</th>
<th>Src/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>D2</td>
<td>(#)</td>
<td></td>
<td>(8/2/1997 ~ 8/2/1997)</td>
<td>0.2</td>
<td>0.4</td>
<td>u</td>
<td>rel3</td>
</tr>
<tr>
<td>D1</td>
<td>D3</td>
<td>(#)</td>
<td></td>
<td>(8/2/1997 ~ 8/3/1997)</td>
<td>0.6</td>
<td>0.8</td>
<td>g.0.5</td>
<td>rel3</td>
</tr>
</tbody>
</table>

Select: 0 > 0.1
Project: \( \beta \leftarrow \Xi \preceq \Xi \) (Data1 \( \forall \), Data2 \( \Box \))

TP-tables

<table>
<thead>
<tr>
<th>tbl_name/s</th>
<th>priority/f</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>rel1</td>
</tr>
<tr>
<td>c</td>
<td>rel2</td>
</tr>
<tr>
<td>c</td>
<td>rel3</td>
</tr>
<tr>
<td>c</td>
<td>rel4</td>
</tr>
</tbody>
</table>

Figure 4: Screen dump of query interface
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 B 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 as seen from Charts (c) and (d).

2. Selection Computations

\(\sigma_C(r)\) and \(\sigma_C(\text{ANN}(r))\) for each type of selection condition \(C\) (i.e., data, temporal, and probabilistic).

We ran three types of experiments with selections involving conditions on data attributes (Charts (e) and (f)), temporal attributes (Charts (g) and (h)), and probabilistic attributes (Charts (i) and (j)), 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 (i) and (j) indicate that performing probabilistic selections on TP-databases that use uniform distributions is faster than on identical TP-databases that use other distributions!

3. Difference and Projection Computations

\(r - r'\), \(\text{ANN}(r) - \text{ANN}(r')\), \(\pi_F(r)\), and \(\pi_F(\text{ANN}(r))\).

Charts (k) and (l) show what happens with Difference, while Charts (m) and (n) show what happens with Projection. The results mirror those in the case of union and intersection.

4. Join Computations

\(\Xi^{by}(r \bowtie_\alpha r')\) and \(\text{ANN}(r) \bowtie_\alpha \text{ANN}(r')\) for each conjunction strategy \(\alpha \in \{ig, pe, nc, in\}\).

We first studied what happens with join under the positive correlation conjunction strategy (Charts (o) and (p)). 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 (q) and (r), 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).

7.1.2 Scalability of TP-Algebra Operations

We studied the performance of two operations in the TP-algebra — selection, and join, as these are two of the most widely used operations. Our interest was to see what happens to the performance of the TP-algebra operations when we execute queries with massive amounts of uncertainty.

Charts (s) and (t) show what happens when we use a mix of distribution functions, and use 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 (s) 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 (t) (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 (s), 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

There has been almost no work to date on the integration of probabilistic databases and temporal reasoning. A notable exception is the work of Dyreson and Snodgrass [13]. We therefore organize this section into three parts — the first part compares our work with that of Dyreson and Snodgrass, the second part compares our work with existing work on probabilistic databases, and the third part compares our work with relevant work on temporal indeterminacy.

8.1 Comparison with Dyreson and Snodgrass

Dyreson and Snodgrass [13] 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 [13] 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 manner described below. In addition, there is an important philosophical difference between our work and theirs — we are adding time to “kosher” probabilistic databases, while they are adding probability to “kosher” temporal databases.

1. **SQL vs. Algebra.** First and foremost, Dyreson and Snodgrass present a version of SQL for temporally indeterminate databases. In contrast, 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.
2. **Base Relations.** In the Dyreson and Snodgrass [13] framework, the base relations are temporal relations. In effect, base relations in [13] may be viewed as special cases of TP-relations where the $C$ and $D$ fields are *atomic time-interval constraints*. In contrast, our framework:

(a) Allows $C$ and $D$ to be arbitrary temporal constraints, thus generalizing their approach. As a consequence, TP-relations can be much more succinct than the base relations used in [13]. This is because a single TP-tuple can often express information about a union of disjoint time intervals.

(b) In [13], no explicit lower/upper bounds are considered; all probabilities used are point probabilities. This is a special case of our framework, and as we have already seen. Recall that in 1854 [7], Boole noticed that we must use probability intervals whenever we are ignorant of the relationship between events.

Conversely, there are some things that can be expressed in the Dyreson-Snodgrass framework [13] which we do not handle — for example, in the current paper, we have assumed tuples have only one indeterminate temporal attribute while [13] allows more than one. Furthermore, we have no analog of correlation credibility or ordering plausibility.

3. **Distribution Functions.** In [13], all PDFs are assumed to be complete. In contrast, in this paper, 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.

4. **Independence Assumptions.** In [13], all indeterminate events are assumed to be independent. This assumption is valid for many applications, and invalid for others. For instance, a transportation plan that involves shipping a packet and then trucking it involves two dependent events — changes in the ship’s arrival time will change the time at which the packet is loaded onto the truck. The independence assumption allows for efficient computation of temporal relationships such as “event $e_1$ occurs before event $e_2$.” In contrast, in our paper, 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 by the user. Our framework supports computation of the probabilistic versions of all 13 operators postulated by Allen [1].

5. **Operators developed.** Our algebra supports a host of operations that do not appear to be supported in the Dyreson-Snodgrass framework [13]. For instance, we provide whole families of compaction methods, combination functions, and compression functions.

6. **Semantics.** We provide formal, model-theoretic descriptions of consistency in our paper for TP-relations, and provide very efficient means to check consistency (linear time) when determinate PDFs are used. When indeterminate PDFs are used, consistency checking is more complex.

7. **Prototype Implementation.** We have implemented our framework on top of the Borland Database Engine, and our experiments complement those of Dyreson and Snodgrass in the sense 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 [13].

### 8.2 Relationship with work in Probabilistic Databases

Dyreson and Snodgrass [13, p.46] stated that they “could not adopt the PDM approach or its successors to support temporal indeterminacy, since there might be several million elements in a set of possible chronons. Representing each alternative with an associated probability is impractical.” This statement is certainly correct if *PDM* is taken to mean the TATA approach. In fact, our experiments on the TATA validate Dyreson and
Snodgrass’ concern — TATA should be used only for theoretical purposes and should not be implemented. However, as we have seen, the TP-Algebra is a much more succinct PDM representation of temporal probabilistic data which can efficiently deal with large sets of chronons.

Kiessling and his group [23, 50, 41] have developed a framework called DUCK for reasoning with uncertainty. They provide an elegant, logical, axiomatic theory for uncertain reasoning in the presence of rules. In the same spirit as Kiessling et al., Ng and Subrahmanian [34, 36] have provided a probabilistic semantics for deductive databases — they assume absolute ignorance, and furthermore, assume that rules are present in the system. In contrast, in our framework, rules are not present; rather, our interest is in extending the relational algebra to capture probabilistic and temporal information. Time is not handled by the DUCK approach.

In an important paper, Lakshmanan and Sadri [32] show how selected probabilistic strategies can be used to extend the previous probabilistic models. Lakshmanan and Shiri [33, 43] have shown how deductive databases may be parametrized through the use of conjunction and disjunction strategies, an approach also followed by Dekhtyar and Subrahmanian [11]. While the ideas behind these frameworks have been used here through the notions of conjunction and disjunction strategies, no notion of time is discussed.

Barbara et al. [3] develop a probabilistic data model and propose probabilistic operators. Their work is based on the assumption that probabilities of compound events can always be precisely determined, an assumption valid for few combination strategies. 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. No temporal data is handled.

Cavallo and Pittarelli [9] propose a model for probabilistic relational databases. In their model, tuples in a probabilistic relation are interpreted using an exclusive or, meaning at most one of the data-tuples is assumed to be present in the underlying classical relation. This is a rather restrictive assumption, and we make no such assumptions. Furthermore, due to the above assumptions, Cavallo and Pittarelli [9] only propose 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 [12] who propose an elegant 1NF approach to handling probabilistic databases. Their paper is a significant improvement to the work of Barbara et al. [3]. In particular, their framework (i) supports having uncertainty about some objects but certain information about others, (ii) uses first normal form which is easy to understand and use, (iii) introduces elegant new operations like conditionalization, and (iv) removes assumptions about deterministic keys prevalent in previous approaches. The 1NF representation used by them is a special case of the annotated representation in this paper — as discussed in Section 8.2 and as pointed out by Dyreson and Snodgrass [13], this representation is not suitable for directly representing temporal indeterminacy. Dey and Sarkar’s approach does not handle time explicitly. 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 have been able to remove through the notion of p-strategies. 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 [12] propose some operations such as conditionalization and Nth-Moment that have no analogs in our paper, and deserve further study.

This paper builds on top of the ProbView system for probabilistic databases[30]. ProbView extends the
classical relational algebra by allowing users to specify in their query, what probabilistic strategy (or strategies) should be used to parametrize the query. ProbView removed the independence assumption from previous works. However, ProbView has no notion of time, and it was noted by Snodgrass [46] 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 its 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.

Several other authors have handled uncertainty in databases through the use of fuzzy sets [15, 24, 38, 39] — as the differences between probabilities and fuzzy sets are well known, we do not address these works extensively here. In addition, uncertainty has been extensively studied in the context of deductive databases and logic programming [11, 24, 28, 32, 31, 33, 34, 36, 37, 42, 52] as well as probabilistic logic [18, 19, 37].

### 8.3 Relationship with work in Temporal Databases

As stated by Dyreson and Snodgrass [13, p.45], “Despite the wealth of research on adding incomplete information to databases, there are few efforts that address incomplete temporal information.” Snodgrass was one of the first to model indeterminate instances in his doctoral dissertation [45] — he proposed the use of a model based on three valued logic. Dutta [17] later proposed a fuzzy logic based approach to handle generalized temporal events — events that may occur multiple times (notice that our framework allows an event to occur multiple times, but each occurrence must be somehow distinguished from other occurrences so that they can be represented by TP-tuples which are not data-identical). This approach is also used by Dubois and Prade [15].

Gadia [20] proposes an elegant model to handle incomplete temporal information as well. He models values that are completely known, values that are unknown but are not to have occurred, values that are known if they have occurred, and values that are unknown even if they occurred. Gadia [20] shows that his model is sound. However, he makes no use of probabilistic information.

An important body of work is that of Koubarakis [26, 27] who proposes the use of constraints for representing temporal data. In this sense, our work is directly related and builds upon Koubarakis’ work. Like us, Koubarakis uses constraints to represent when an event occurs. Koubarakis’ 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 [26, 27] can explicitly encode in his tuples.

Another important body of work is that of Brusoni et al.[8] 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 [10].
9 Conclusions and Future Work

There are a large variety of applications where there is uncertainty about when certain real-world events occurred, or are predicted to occur. 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. The same is true of weather applications where predictions about the likelihood of rain in a certain time interval also has probabilistic attributes. A variety of other important applications involving uncertainty about when events occur have been identified in an important paper by Dyreson and Snodgrass[13].

The only previous work whose explicit goal was to incorporate uncertainty into temporal databases is due to Dyreson and Snodgrass [13]. 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 [13].

- 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 [13].
- 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 parametrize 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 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 [53].
- 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.

This is the first in a long term research effort we have on probabilistic temporal databases. Our long term research program involves enhancing the framework of TP-programs in the following ways.

1. First, due to space reasons, we have not been able to study query optimization in this paper. We are working on establishing a set of rewrite rules, and a set of cost models for TP-databases. Preliminary test runs with optimized versions of some queries show far superior performance than indicated in the experiments in this paper.

2. Second, we are working on the problem of incremental view maintenance in TP-databases. This is an important problem because in many TP applications (such as stock applications and transportation applications), updates occur all the time. Handling these updates efficiently is critical.

3. A third important point is that of integrity constraints. What does it mean for a database state to satisfy an integrity constraint, when in fact, the contents of the database is uncertain and reflects a set of possible states, together with an associated probability distribution.

4. A fourth major research topic is on indexing TP-databases. While extensive work has been carried out on indexing constraints, and indexing relational databases, the problem of indexing TP-cases introduces new twists when we wish to support efficient probabilistic queries.

5. A fifth major research topic is that of supporting probabilistic aggregate operations. For instance, this is needed to support queries of the form *Find the ten most probable events to occur at time t.*

Concurrently with the above, we are involved in continuously improving our implementation through the addition of a variety of features.

**Acknowledgements**

Different parts of this work were supported by the Army Research Office under Grants DAAH-04-95-10174, 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, and by a grant from Lockheed Martin Advanced Technology Laboratories.

**References**


[33] 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.


## Appendix: Notation tables

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
<th>§</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>Time unit e.g. (day, {1, \ldots, 31})</td>
<td>2.1</td>
</tr>
<tr>
<td>$H = T_1 \subseteq \cdots \subseteq T_n$</td>
<td>Linear hierarchy of time units</td>
<td>2.1</td>
</tr>
<tr>
<td>$t = (v_1, \ldots, v_n)$</td>
<td>Time point e.g. (2/3/1996)</td>
<td>2.1</td>
</tr>
<tr>
<td>$t &lt;_H t'$</td>
<td>Time point $t$ occurs before $t'$</td>
<td>2.1</td>
</tr>
<tr>
<td>$\tau$</td>
<td>Calendar e.g. Gregorian</td>
<td>2.1</td>
</tr>
<tr>
<td>$\text{next}_\tau(t)$</td>
<td>Next, consecutive time point after $t$</td>
<td>2.1</td>
</tr>
<tr>
<td>$C, D$</td>
<td>Temporal constraints e.g. ($t_1 \sim t_2$)</td>
<td>2.2</td>
</tr>
<tr>
<td>$S_{\tau} = \text{sol}(t_\tau \sim t_{E'})$</td>
<td>Set of all valid time points over $\tau$</td>
<td>2.2</td>
</tr>
<tr>
<td>$\text{sol}(C) = S \subseteq S_{\tau}$</td>
<td>Solution set for temporal constraint $C$</td>
<td>2.2</td>
</tr>
<tr>
<td>$\text{pdf}(D, t_j)$</td>
<td>Determines probability $\rho_j$ for $t_j \in \text{sol}(D)$</td>
<td>2.3</td>
</tr>
<tr>
<td>$\delta$</td>
<td>Distribution function (names a family of PDFs)</td>
<td>2.3</td>
</tr>
<tr>
<td>$\delta = u, g, g_c, b, po$</td>
<td>Uniform, geometric, complete $g$, binomial, Poisson</td>
<td>2.3</td>
</tr>
<tr>
<td>$pt = \langle D, L, U, \delta \rangle$</td>
<td>P-tuple; determines $[L_t, U_t]$ for each $t \in \text{sol}(D)$</td>
<td>2.4</td>
</tr>
<tr>
<td>$\text{sl}(e), \text{end}(e)$</td>
<td>Instantaneous start/end events which bounding event $e$</td>
<td>2.4</td>
</tr>
<tr>
<td>$\oplus_\alpha, \oplus_\beta$</td>
<td>Probabilistic conjunction/disjunction strategy</td>
<td>2.5</td>
</tr>
<tr>
<td>$\alpha / \beta = ig, pc, ne, in$</td>
<td>Ignorance, positive/negative correlation, independence</td>
<td>2.5</td>
</tr>
<tr>
<td>$\chi(S)$</td>
<td>Combination function; combines each $[L_t, U_t] \in S$</td>
<td>2.6</td>
</tr>
<tr>
<td>$\chi_{eq}$</td>
<td>Optimistic equity combination function</td>
<td>2.6</td>
</tr>
<tr>
<td>$\gamma_i = \langle C_i, D_i, L_i, U_i, \delta_i \rangle$</td>
<td>TP-case; $C_i$ may be # if $\text{sol}(C_i) = \text{sol}(D_i)$</td>
<td>3.1</td>
</tr>
<tr>
<td>$\gamma = {\gamma_1, \ldots, \gamma_n}$</td>
<td>TP-case statement; each TP-case $\gamma_i \in \gamma$</td>
<td>3.1</td>
</tr>
<tr>
<td>$d = (d_1, \ldots, d_k)$</td>
<td>Data tuple over relational schema $A = (A_1, \ldots, A_k)$</td>
<td>3.2</td>
</tr>
<tr>
<td>$tp = (d, \gamma) \in r$</td>
<td>TP-tuple in TP-relation $r$</td>
<td>3.2</td>
</tr>
<tr>
<td>$\mathcal{P}(d)$</td>
<td>Manifest projection of data tuple $d$; removes $d.H$</td>
<td>3.2</td>
</tr>
<tr>
<td>$d.H</td>
<td>d'.H$</td>
<td>Hidden list concatenation; merge of field-value pairs</td>
</tr>
<tr>
<td>$r[tp]$</td>
<td>Multiset of all TP-tuples in $r$ which are data-identical to $tp$</td>
<td>3.4</td>
</tr>
<tr>
<td>$\text{dom}(A)$</td>
<td>Domain of relational schema $A$; $\text{dom}(A_1) \times \cdots \times \text{dom}(A_k)$</td>
<td>3.4</td>
</tr>
<tr>
<td>$I_{A,\tau}(d, t)$</td>
<td>TP-Interpretation; probability that $d$’s event is true at time $t$</td>
<td>3.4</td>
</tr>
<tr>
<td>$I_{A,\tau}(d, D)$</td>
<td>Probability assigned by $I_{A,\tau}$ to temporal constraint $D$</td>
<td>3.4</td>
</tr>
<tr>
<td>$I_{A,\tau} \models tp$</td>
<td>TP-Interpretation $I_{A,\tau}$ satisfies TP-tuple $tp$</td>
<td>3.4</td>
</tr>
<tr>
<td>$\text{ANN}(tp), \text{ANN}(r)$</td>
<td>Annotated relations for $tp$ and $r$</td>
<td>4.1</td>
</tr>
<tr>
<td>$\text{at} = (d, \gamma, L_i, U_i)$</td>
<td>Annotated tuple for $tp = (d, \gamma)$ at time $t$</td>
<td>4.1</td>
</tr>
<tr>
<td>$\text{ANN}(r)[d, t]$</td>
<td>Multiset of all $\text{at} \in \text{ANN}(r)$ where $(\text{at}, d = d \land \text{at}, t = t)$</td>
<td>4.2</td>
</tr>
<tr>
<td>Symbol</td>
<td>Meaning</td>
<td>§</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------------------------------------------</td>
<td>----</td>
</tr>
<tr>
<td>$\kappa_\chi(ANN(r))$, $\kappa_\phi(ANN(r))$</td>
<td>Combination function/p-strategy based compaction</td>
<td>5.1</td>
</tr>
<tr>
<td>$ANN(r) \cap ANN(r')$</td>
<td>Multiset intersection</td>
<td>5.2</td>
</tr>
<tr>
<td>$ANN(r) \cap_\chi ANN(r')$</td>
<td>Denotes $\kappa_\chi(ANN(r) \cap ANN(r'))$</td>
<td>5.2</td>
</tr>
<tr>
<td>$ANN(r) \cup ANN(r')$</td>
<td>Multiset union</td>
<td>5.3</td>
</tr>
<tr>
<td>$ANN(r) \cup_\chi ANN(r')$</td>
<td>Denotes $\kappa_\chi(ANN(r) \cup ANN(r'))$</td>
<td>5.3</td>
</tr>
<tr>
<td>$\sigma_C(ANN(r))$</td>
<td>Selection of condition $C$</td>
<td>5.4</td>
</tr>
<tr>
<td>$ANN(r) - ANN(r')$</td>
<td>Difference</td>
<td>5.5</td>
</tr>
<tr>
<td>$ANN(r) \times_\alpha ANN(r')$</td>
<td>Cartesian product under $\alpha$</td>
<td>5.6</td>
</tr>
<tr>
<td>$\pi_\mathcal{F}(ANN(r))$</td>
<td>Multiset projection of projectable field list $\mathcal{F}$</td>
<td>5.7</td>
</tr>
<tr>
<td>$ANN(r) \bowtie_\alpha ANN(r')$</td>
<td>Join; $\pi_\mathcal{F}(\sigma_C(ANN(r) \times_\alpha ANN(r')))$</td>
<td>5.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
<th>§</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N(r), N(tp)$</td>
<td>Number of TP-cases in $r/tp$</td>
<td>6.1</td>
</tr>
<tr>
<td>$\Xi^{sd}(r), \Xi^{u}(r), \Xi^{hy}(r)$</td>
<td>Same distribution/uniform/hybrid TP-compression</td>
<td>6.1</td>
</tr>
<tr>
<td>$\kappa_\chi(r), \kappa_\phi(r)$</td>
<td>Combination function/p-strategy based compaction</td>
<td>6.2</td>
</tr>
<tr>
<td>$r \cap r'$</td>
<td>Multiset intersection</td>
<td>6.3</td>
</tr>
<tr>
<td>$r \cap_\chi r'$</td>
<td>Denotes $\kappa_\chi(r \cap r')$</td>
<td>6.3</td>
</tr>
<tr>
<td>$r \cup r'$</td>
<td>Multiset union</td>
<td>6.4</td>
</tr>
<tr>
<td>$r \cup_\chi r'$</td>
<td>Denotes $\kappa_\chi(r \cup r')$</td>
<td>6.4</td>
</tr>
<tr>
<td>$C^\parallel = \text{TP-filter}(\gamma_t, C)$</td>
<td>TP-filter of $\gamma_t$ w.r.t. probabilistic condition $C$</td>
<td>6.5</td>
</tr>
<tr>
<td>$\sigma_C(tp), \sigma_C(r)$</td>
<td>Selection of condition $C$ on $tp/r$</td>
<td>6.5</td>
</tr>
<tr>
<td>$r - r'$</td>
<td>Difference</td>
<td>6.6</td>
</tr>
<tr>
<td>$r \times_\alpha r'$</td>
<td>Cartesian product under $\alpha$</td>
<td>6.7</td>
</tr>
<tr>
<td>$\pi_\mathcal{F}(r)$</td>
<td>Multiset projection of field list $\mathcal{F}$</td>
<td>6.8</td>
</tr>
<tr>
<td>$r \bowtie_\alpha r'$</td>
<td>Join; $\pi_\mathcal{F}(\sigma_C(r \times_\alpha r'))$</td>
<td>6.9</td>
</tr>
</tbody>
</table>

B Appendix: Experimental results
Union when nTimePoints = 16

Union when nTuples = 1000

c
(d)
Selection (data condition) when nTimePoints = 16

Selection (data condition) when nTuples = 1000
Selection (temporal condition) when nTimePoints = 16

Selection (temporal condition) when nTuples = 1000
Selection (probabilistic condition) when nTimePoints = 16

Selection (probabilistic condition) when nTuples = 1000
Join when nTimePoints = 16, conjStategy = pc

Join when nTuples = 1000, conjStategy = pc
Join when nTimePoints = 16, delta = mix

Join when nTimePoints = 1000, delta = mix
C Appendix: Proofs of results

Proof of Proposition 1.

Let \( \tau \) be a calendar and \((name_1,\ldots,name_k)\) be the list of the names of its time units.

To prove this proposition it suffices to show that (i) every atomic time-value constraint can be represented as a time-interval constraint and (ii) every atomic time-interval constraint can be represented as a time-value constraint.

We do just that below:

1. For every atomic time-interval constraint \( C \) there exists a time-value constraint \( C' \) such that \( \text{sol}(C) = \text{sol}(C') \).

   Let \( C = (t_1 \sim t_2) \) be an atomic time-interval constraint. Let \( t_1 = (v_1,\ldots,v_k) \) and \( t_2 = (v'_1,\ldots,v'_k) \).

   Now, let \( \text{sol}(C) = \{s_1,\ldots,s_m\} \).

   Let \( s = (v'_1,\ldots,v'_k) \) be a time point. We construct a time-value constraint \( C_s \) as follows:

   \[
   C_s = (name_1 = v'_1 \land name_2 = v'_2 \land \ldots \land name_k = v'_k)
   \]

   By definition of solution set for a temporal constraint \( \text{sol}(C_s) = \{s\} \). But then for a time-value constraint

   \[
   C' = C_{s_1} \lor C_{s_2} \lor \ldots \lor C_{s_k}
   \]

   the solution set is defined as the union of solution sets of \( C_{s_i} (1 \leq i \leq k) \), i.e.:

   \[
   \text{sol}(C') = \text{sol}(C_{s_1}) \cup \ldots \cup \text{sol}(C_{s_k}) = \{s_1\} \cup \ldots \cup \{s_k\} = \{s_1,\ldots,s_k\} = \text{sol}(C).
   \]

   \textbf{Note:} \( C' \) constructed as described above is going to be a very large constraint. In practice it is always possible to construct much more compact time-value constraints for representing a time period.

2. For every atomic time-value constraint \( C \) there exists a time-interval constraint \( C' \) such that \( \text{sol}(C) = \text{sol}(C') \).

   Let \( C = (name_i;opv_i) \) be an atomic time-value constraint. We construct an equivalent time-interval constraint as follows.

   Let \( t \) be a timepoint and let \( C_t \) be a time-interval constraint defined as follows:

   \[
   C_t = (t \sim t)
   \]

   By definition of the solution set of a temporal constraint, \( \text{sol}(C_s) = \{s\} \).

   Now, let \( \text{sol}(C) = \{t_1,\ldots,t_m\} \). The solution set of a time-interval constraint \( C' \) defined as

   \[
   C' = C_{t_1} \lor C_{t_2} \lor \ldots \lor C_{t_m}
   \]

   will be equal to

   \[
   \text{sol}(C') = \text{sol}(C_{t_1}) \cup \ldots \cup \text{sol}(C_{t_m}) = \{t_1\} \cup \ldots \cup \{t_m\} = \{t_1,\ldots,t_m\} = \text{sol}(C)
   \]

   \textbf{Note:} \( C' \) constructed as described above is going to be a very large constraint. In practice it is always possible to construct much more compact time-value constraints for representing a time period. \qed
Proof of Proposition 2.
If \( S \) is such that \( \cap_{[L, U] \in S} \{ L, U \} \neq \emptyset \) then the values of all functions defined in the example 2.5 will coincide on \( S \), as they are all defined to be equal to the intersection. Clearly, \( \cap_{[L, U] \in S} \{ L, U \} \) satisfies identity as \( [L, U] \cap [L, U] = [L, U] \). Also, since \( \cap_{[L, U] \in S} [L, U] = [\max_{[L, U] \in S} L, \min_{[L, U] \in S} U] \) and \( \cap_{[L, U] \in S} L, \min_{[L, U] \in S} U \leq \min_{[L, U] \in S} L(1) \) the Bottomline is satisfied. We also notice that when \( S = \{ [I, U] \}, \cap_{[L, U] \in S} [L, U] \neq \emptyset \), i.e. identity need not be considered when \( \cap_{[L, U] \in S} [L, U] = \emptyset \).

Let now \( \cap_{[L, U] \in S} [L, U] = \emptyset \). We prove Bottomline for all six functions:

- **Optimistic Equity** \( \chi_{eo}(S) = [\max_{[L, U] \in S} L, \max_{[L, U] \in S} U] \) i.e. \( \chi_{eo} \) satisfies Bottomline.
- **Enclosing Equity** \( \chi_{ee}(S) = [\min_{[L, U] \in S} L, \max_{[L, U] \in S} U] \) i.e. \( \chi_{ee} \) satisfies Bottomline.
- **Pessimistic Equity** \( \chi_{ep}(S) = [\min_{[L, U] \in S} L, \min_{[L, U] \in S} U] \) i.e. \( \chi_{ep} \) satisfies Bottomline.
- **Rejecting Equity** \( \chi_{er}(S) = [0, 0] \) i.e. \( \chi_{er} \) satisfies Bottomline.
- **Skeptical Equity** \( \chi_{es}(S) = [0, 1] \).
- **Quasi-Independence Equity** \( \chi_{ei}(S) = [\Pi_{[L, U] \in S} L, \Pi_{[L, U] \in S} U] \) \( \leq [\max_{[L, U] \in S} L, 1] \).

Proof of Theorem 1.

- \( r \) is compact \( \implies \) \( ANN(r) \) is compact.

  As \( r \) is compact, for each data tuple \( d \) and timepoint \( t \) there is at most one tp-tuple \( t_p = (d, \gamma) \in r \), \( \gamma = \gamma_1, \ldots, \gamma_k, \gamma_i = (C_i, D_i, L_i, U_i, \delta_i) \), for \( 1 \leq i \leq k \) such that \( t \in sol(C_1 \lor \ldots \lor C_k) \).

  Let \( d \) be some data tuple such that there exists a tuple \( t_p = (d, \gamma) \in r \). We show that \( ANN(r[t_p]) \) is compact. Let \( r[t_p] = \{ t_p, t_{p_2}, \ldots, t_{p_m} \} \). Let \( C_j \) be the disjunction of all \( C\)-temporal constraints in \( t_{p_j} \). As \( r \) is compact, we know that \( (\forall 1 \leq i \neq j \leq m) (sol(C^j) \cap sol(C^i) = \emptyset) \). Let \( t \) be a timepoint in \( sol(C^1 \lor \ldots \lor C^m) \). Then there exists a unique \( 1 \leq h \leq m \) such that \( t \in sol(C^h) \). Consider tp-tuple \( t_{p_h} = (d, \gamma') \) where \( \gamma' = \gamma'_1, \ldots, \gamma'_k \) and \( \gamma'_i = (C_i, D_i, L_i, U_i, \delta_i) \) for \( 1 \leq i \leq k \). By the definition of a TP-tuple, \( \forall 1 \leq i \neq j \leq k (sol(C_i) \cap sol(C_j) = \emptyset) \). This means that for each \( t \in sol(C_1 \lor \ldots \lor C_k) \) there will be only one annotated tuple \( (d, t, L_{t}, U_{t}) \) added to \( ANN(t_{p_h}) \), i.e., \( ANN(t_{p_h}) \) has at most one tuple \( (d, t, L_{t}, U_{t}) \) for each timepoint \( t \). The latter means that \( ANN(r[t_p]) \) is compact, since for two tuples \( t_{p'}, \ t_{p''} \) \( r[t_p] \), and two tuples \( a_t' = (d', \gamma', U', U'') \in ANN(t_{p'}) \) and \( a_t'' = (d, t^0, t'^0, U'') \in ANN(t_{p''}) \) it will never be the case that \( t' = t'' \). Compactness of \( ANN(r[t_p]) \) for any \( t \in r \) yields the compactness of \( ANN(r) \), as for any two not data-identical tuples \( t_p \) and \( t'_{p'} \) in \( r \), no two tuples \( a_t \in ANN(r[t_p]) \) and \( a_t' \in ANN(r[t'_{p'}]) \) will be data-identical.

- \( ANN(r) \) is compact \( \implies r \) is compact.

  Let \( r \) be a tp-relation. Consider an arbitrary annotated tuple \( a_t = (d, t, L_{t}, U_{t}) \in ANN(r) \). As \( ANN(r) \) is compact, \( ANN(r[d, t]) = \{ at \} \), i.e. no other annotated tuple in \( ANN(r) \) is data-time identical to \( at \). By definition of the operator \( ANN(\cdot) \), there exists a tp-tuple \( t \in r \) such that, \( t = (d, \gamma), \gamma = \gamma_1, \ldots, \gamma_k, \gamma_i = (C_i, D_i, L_i, U_i, \delta_i) \), for \( 1 \leq i \leq k \) and \( t \in ANN(t_{p_h}) \), i.e. \( t \in sol(C_1 \lor \ldots \lor C_k) \).

  As \( ANN(r[d, t]) = \{ at \} \), for any other tp-tuple \( t' = (d, \gamma') \in r \) such that \( \gamma' = \gamma'_1, \ldots, \gamma'_k, \gamma'_i = (C'_i, D'_i, L'_i, U'_i, \delta'_i) \) we will have \( t \notin sol(C'_1 \lor \ldots \lor C'_k) \). Therefore, by definition 6.6, \( r \) is a compact tp-relation. \( \square \)
Proof of Theorem 2. We show a stronger result, viz. that if \( I \models r \) then also \( I \models \tmop{ANN}(r) \). Let \( tp = (d, \gamma) \in r \) be some tp-tuple and \( at = (d, t, L, U) \in \tmop{ANN}(r) \). Since \( I \models r, I \models tp \). We need to show that \( I \models at \).

Let \( \gamma = \gamma_1, \ldots, \gamma_k \), and \( \gamma_i = \langle C_i, D_i, L_i, U_i, \delta_i \rangle \) for \( 1 \leq i \leq k \). Since \( at = (d, t, L, U) \in \tmop{ANN}(r) \), there exists some \( 1 \leq j \leq k \) such that, \( t \in \tmop{sol}(C_j) \) and \( [L, U] = [\delta(D_j, t)L_j, \delta(D_j, t)U_j] \).

Since \( I \models tp \), \( I \models (d, D_j, [L_j, U_j, \delta_j]) \). By the definition of satisfaction, \( I(d, D_j) \in [L_j, U_j] \) and \( I(d, t) = \delta_j(D_j, t)I_d(d, D_j) \). But then \( L = \delta_j(D_j, t)L_j \leq \delta_j(D_j, t)I(d, D_j) = I(d, t) = \delta_j(D_j, t)U_j = U \), i.e., \( I(d, t) \in [L, U] \). From the latter it follows that \( I \models (d, t, L, U) = at \).

\[ \Box \]

Proof of Theorem 3. Let \( I_{A, r}(d, t) \) be computed as follows. As \( r \) is compact, all tuples of the form \( (d_1, \ldots, d_r) \)
\( \in \tmop{ANN}(r) \) are derived from \( \tmop{ANN}(tp) \) where \( tp = (d, \gamma) \) where \( \gamma = \langle C_1, D_1, L_1, U_1, \delta_1 \rangle, \ldots, \langle C_n, D_n, L_n, U_n, \delta_n \rangle \).

Let \( \tmop{ANN}(tp) \) consist of \( (d_1, t_1, L_1, U_1), \ldots, (d_k, t_k, L_k, U_k) \). We know that \( \Sigma_{i=1}^k L_i \leq 1 \). Without loss of generality, suppose \( t \) is a solution of \( C_i \) for some \( 1 \leq i \leq m \). Clearly, exactly one such \( i \) exists. Let \( \{t'_1, \ldots, t'_{r+1}, \ldots, t'_{r+1} \} \) be all other solutions of \( D_i \), and let \( \tmop{sol}(C_i) = \{t'_1, \ldots, t'_{r+1} \} \). For all \( 1 \leq i \leq r \), \( (d, t', L_i, U_i) \) is in \( \tmop{ANN}(r) \). We know that \( L_i \leq L + L'_1 + \cdots + L'_r \leq U_i \). We now construct an interpretation \( I_{A, r} \) that satisfies \( r \) and which assigns \( L \) to \( I_{A, r}(d, t) \) as follows. We know that \( x \cdot I_{A, r}(d, y) = \delta_i(D_i, y) \) for some \( x \geq 0 \). By setting \( I_{A, r}(d, t) = L \), we know that \( x = \frac{\delta_i(D_i, t)}{L} \). Therefore, we may set
\[ I_{A, r}(d, t') = \delta_i(D_i, t') \times \frac{L}{\delta_i(D_i, t)}. \]

The same procedure may be used to extend \( I_{A, r} \) to other tuples that are not data-identical to \( (d, -, -, -) \).

\[ \Box \]

Proof of Theorem 4. Similar to the proof of Theorem 3.

\[ \Box \]

Proof of Proposition 3.

Suppose \( r \) is compact. As \( r \) contains no distinct data-identical tuples, \( r \) is consistent iff all tp-tuples in \( r \) are individually consistent. 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
\[ (L_1 + \cdots + L_n) \leq \min(1, \Sigma_{i=1}^n L_i \times \Sigma_{t \in \tmop{sol}(C_i)} \delta_i(D_i, t)). \]

If this condition holds, then the probabilistic interpretation \( I_{A, r} \) defined as follows satisfies \( tp \).
\[ I_{A, r}(d, t) = \begin{cases} L_i \times \delta_i(D_i, t) & \text{if } t \in \tmop{sol}(C_i), 1 \leq i \leq n \\ 0 & \text{otherwise.} \end{cases} \]

As all the distributions \( \delta_i \) are determinate, it is easy to see that this check is linear in the size of the relation \( r \).

\[ \Box \]

Proof of Proposition 4.

Let \( \gamma \) be a combination function and let \( \tmop{ANN}(r) \) be an annotated relation. We show that \( \kappa_\gamma \) satisfies the properties of compaction operations.

1. Compactness. By definition 5.2 \( \kappa_\gamma(\tmop{ANN}(r)) \) contains only one annotated tuple for each pair \( (d, t) \) where \( d \) is data tuple and \( t \) is a timepoint. Therefore, \( \kappa_\gamma(\tmop{ANN}(r)) \) is compact.

2. No Fooling Around. Let \( \tmop{ANN}(r) \) be compact. Then for any data tuple \( d \) and timepoint \( t \), \( \tmop{ANN}(r)[d, t] \) will contain at most one tuple. Let \( \tmop{ANN}(r)[d, t] = \{ (d, t, L, U) \} \). Then, by definition 5.2, \( \kappa_\gamma h(\tmop{ANN}(r)) \)
will contain a tuple \((d, t, \chi([L_i, U_i]))\). As \(\chi\) is a combination function, it satisfies the identity property of combination functions, i.e. \(\chi([d, t, L_i, U_i]) = [L_i, U_i]\), i.e., \((d, t, L_i, U_i) \in \kappa_{\chi}(AN N(r))\) or \(AN N(r) \subseteq \kappa_{\chi}(AN N(r))\).

To prove that \(\kappa_{\chi}(AN N(r)) \subseteq AN N(r)\) we observe that by definition 5.2, the only tuples in \(\kappa_{\chi}(AN N(r))\) are those inserted there by applying \(\chi\) to the sets \([L_i, U_i]\) of probabilistic intervals from \(AN N(r)[d, t]\) equivalence classes. As we have shown above, all such tuples will be the tuples from the original relation \(AN N(r)\).

3. Conservativeness. We notice that if there exists a tuple \((d, t, L_i) \in \kappa_{\chi}(AN N(r))\), then by definition 5.2 there had to be a nonempty set \(AN N(r)[d, t] \subseteq AN N(r)\), which means that for any such tuple in \(\kappa_{\chi}(AN N(r))\) there is always at least one tuple \((d, t, L_i', U_i')\) in \(AN N(r)\).

Proof of Theorem 5. Let \(\chi\) be a combination function and \(AN N(r)\) be an annotated relation. Let \((d, t, L', U') \in AN N(r)\). Then \(AN N(r)[d, t] \neq \emptyset\). Let \(AN N(r)[d, t] = \{a_1, \ldots, a_k\}\), \(a_i = (d, t, L_i', U_i')\) for \(1 \leq i \leq k\). By definition of combination function-based compaction, \(\kappa_{\chi}(AN N(r))\) will contain the tuple \((d, t, L', U')\) where \([L, U] = \chi([L_i', U_i'], \ldots, [L_k', U_k'])\).

Proof of Proposition 5. We need to show that \(\chi_{\ominus}\) satisfies the three axioms defining compaction operators.

1. Compactness. By the definition of a p-strategy based compaction, for any given pair \((d, t)\), \(\kappa_p\) computes \(AN N(r)[d, t]\) consisting of all tuples in \(AN N(r)\) of the form \((d, t, -, -, -)\). All tuples in \(AN N(r)[d, t]\) are combined into one by the construction of the definition of a p-strategy based compaction.

2. No Fooling Around. If \(AN N(r)\) is compact, then for all \((d, t)\), \(AN N(r)\) contains at most one tuple of the form \((d, t, -, -, -)\). In this case, the definition of \(\kappa_p\) returns that tuple unchanged.

3. Conservativeness. Suppose \(\kappa_p(AN N(r))\) contains a tuple of the form \((d, t, -, -, -)\). Then it is easy to see that by the definition a of p-strategy based compaction there must exist at least one tuple in \(AN N(r)\) of the form \((d, t, -, -, -)\).

Proof of Theorem 6. We need to show that \(\Xi^{sd}(r)\), \(\Xi^u(r)\) and \(\Xi^{by}(r)\) are all TP-compression functions.

To see that \(\Xi^{sd}(r)\) is a TP-compression function, we must demonstrate that \(N(\Xi^{sd}(r) \leq N(r)\) and that there exists a bijection, \(\phi\) from \(AN N(r)\) to \(AN N(sd(r))\). In fact we will show that a stronger statement holds: \(AN N(r) = AN N(sd(r))\).

1. \(N(\Xi^{sd}(r)) \leq N(r)\).

Let \(r = \{t_{p_1}, \ldots, t_{p_s}\}\). Clearly, \(N(r) = N(t_{p_1}) + \cdots + N(t_{p_s})\).

By definition of \(\Xi^{sd}\), for every tp-tuple \(tp \in r\), exactly one tp-tuple \(tp'\) is added to \(\Xi^{sd}(r)\). Therefore \(|r| = |\Xi^{sd}(r)|\). Let now \(\Xi^{sd}(r) = \{tp'_1, \ldots, tp'_s\}\) where \(tp'_i = \Xi^{sd}(t_{p_i})\), \(1 \leq i \leq s\). As \(N(\Xi^{sd}(r)) = N(t_{p'_1}) + \cdots + N(t_{p'_s})\), it suffices to show that for all \(1 \leq i \leq s\), \(N(tp'_i) \leq N(t_{p_i})\).

The latter statement is clearly true as by definition of \(\Xi^{sd}\) either \(tp_i = t_{p'_i}\) (in the case when no tp case statements were deleted) or, for every two tp-case statements deleted from the tp-tuple \(tp_i\) during its conversion into \(tp'_i\) only one tp-case statement is added.

2. \(AN N(r) = AN N(\Xi^{sd}(r))\).

Let \(tp \in r\). We will show that \(AN N(tp) = AN N(\Xi^{sd}(tp))\).

Let \(tp' = \Xi^{sd}(tp)\). Let \(tp = (d, \gamma), \gamma = \gamma_1, \ldots, \gamma_m, \gamma_i \in \langle C_i, D_i, L_i, U_i, \delta_i \rangle, 1 \leq i \leq n\) and let \(tp'_i = (d, \gamma'_i), \gamma'_i = \gamma'_i_1, \ldots, \gamma'_i_m, \gamma'_i_i \in \langle C'_i, D'_i, L'_i, U'_i, \delta'_i \rangle, 1 \leq i \leq m\).

Consider some tp-case \(\gamma'_i\) from \(tp'_i\). By the definition of \(\Xi^{sd}\), one of two cases is possible:
(a) There exists some \( 1 \leq k \leq n \), \( \gamma_j^* = \gamma_k \) (i.e., during the application of \( \Xi^{s_1} \), tp-case \( \gamma_k \) did not change). In this case clearly \( AN_N((d, \gamma_k)) = AN_N((d, \gamma_j)) \).

(b) \( \gamma_j^* \) is not equal to any \( \gamma_i \in \gamma \), but, there exist \( 1 \leq k_1, \ldots, k_q \leq n \) such that \( D_{k_1} = \ldots = D_{k_q} = D_j \), \( L_{k_1} = \ldots = L_{k_q} = L_j \), \( U_{k_1} = \ldots = U_{k_q} = U_j \), \( \delta_{k_1} = \ldots = \delta_{k_q} = \delta_j \), and \( C_{k_1} \cup \ldots \cup C_{k_q} = C_j \). Let now \( at = (d, t, L, U) \in AN_N(d, (\gamma_1, \ldots, \gamma_{k_2})) \). Clearly, there exists some \( 1 \leq h \leq q \), such that \( t \in sol(C_{k_h}) \) and \( [L, U] = [\delta_{k_h}(D_{k_h}, t) \cdot L_{k_h}, \delta_{k_h}(D_{k_h}, t) \cdot U_{k_h}] \). Since \( t \in sol(C_{k_h}) \), \( t \in sol(C_{k_1} \cup \ldots \cup C_{k_q}) = sol(C_j) \). Therefore, \( at' = (d, t, \delta_{k_1}(D'_j, t) \cdot L'_j, \delta_{k_2}(D'_j, t) \cdot U'_j) \in AN_N(d, \gamma_j) \). But since \( D_{k_h} = D'_j \), \( L_{k_h} = L_j \), \( U_{k_h} = U'_j \) and \( \delta_{k_h} = \delta_j \), \( [L, U] = [\delta_{k_1}(D'_j, t) \cdot L'_j, \delta_{k_1}(D'_j, t) \cdot U'_j] \) and therefore \( at = at' \). Hence, \( AN_N(d, (\gamma_1, \ldots, \gamma_{k_2})) \subseteq AN_N(AN_N(d, \gamma_j)) \). The proof that \( AN_N(d, (\gamma_1, \ldots, \gamma_{k_2})) \supseteq AN_N(AN_N(d, \gamma_j)) \) is analogous. This establishes that \( AN_N(d, (\gamma_1, \ldots, \gamma_{k_2})) = AN_N(AN_N(d, \gamma_j)) \).

Combining the results from above together we conclude that \( AN_N(tp) = AN_N(\Xi^{s_1}(tp)) \).

To show that \( AN_N(tp) \subseteq AN_N(\Xi^{s_1}(tp)) \) it suffices to notice that for any \( 1 \leq i \leq n \) and tp-case \( \gamma_i \) there exists such \( 1 \leq j \leq m \) and tp-case \( \gamma_j^* \) that \( sol(C_i) \subseteq sol(C_j) \).

The proofs for \( \Xi^u(r) \) and \( \Xi^{by}(r) \) are similar.

Proof of Lemma 1.
Similar to the proof of Proposition 4.

Proof of Theorem 7.

Let us look at the relation \( r'' \) which is returned by \textbf{Compute-Compaction} \((r, \chi)\). The main loop of the algorithm (lines 04.--23.) works in the following way:

- In lines 05.--08. of the algorithm the initial relation \( r \) is broken into the equivalence classes by the relation of data-identity. Lines 05.--06. select the next unprocessed equivalence class, and lines 07.--08. prepare the data for the next step. In particular, \( \Gamma_I \) (and initially \( \Gamma_S \)) are set to be equal to the set of all tp-cases found in the tp-tuples of the current equivalence class.

- The \textbf{foreach} loop in lines 09.--12. then separates all timepoints which are in solution of some tp-case \( \gamma_i \in \Gamma_I \) into two categories: timepoints that are referred to \textbf{only} by \( \gamma \) and timepoints referred to by more than one tp-case. The timepoints referred to only by one tp-case are handled by line 11. in which a new tp-case that contains only such timepoints from \( \gamma_i \) and adds it to the final tp-tuple.

- Line 13. collects together in one temporal constraint \( C_S \) all the timepoints that are referred to by more than one original tp-case.

- In the \textbf{foreach} loop in lines 15.--22. each timepoint \( t \) from \( sol(C_S) \) gets processed. Lines 17.--20. collect together the set \( X \) of all intervals \([L, U]\) which are the probabilities associated with the timepoint \( t \) by all original tp-cases which refer to \( t \).

- In line 21. the combination function \( \chi \) is applied to the set \( X \) to obtain the interval \([l''_t, u''_t]\), which will become the part of the tp-case \((l, t, l''_t, u''_t, u)\) which is added to the final tp-tuple \( tp'' \).

- In line 23. the tp-tuple \( tp'' \) constructed for the current equivalence class is added to the resulting relation \( r'' \).
Now, let some annotated tuple $at = (d, t, L, U) \in ANN(r'^d)$. Then there exists a tp-tuple $tp \in r'^d$, such that $at \in ANN(tp)$. Let us examine the values of $L$ and $U$.

Since $at \in ANN(tp), tp = (d, \gamma), \gamma = \gamma_1, \ldots, \gamma_n, \gamma_i = \langle C_i, D_i, L_i, U_i, \delta_i \rangle, 1 \leq i \leq n$ and there exists $1 \leq j \leq n, t \in sol(C_j)$ and $[L, U] = [\delta_j(D_j, t) \cdot L_j, \delta_j(D_j, t) \cdot U_j]$.

Let us look at how tp-case $\gamma_i$ could have been added to $tp$. From the analysis of the algorithm above, it is clear that there are only two possibilities:

1. $\gamma_i$ had been added to $tp$ in line 11. Let us look at the equivalence class of $tp$ in $r$ (although $tp$ itself is not in $r$ it must be data-identical to some tp-tuples of $r$ and hence $r[tp]$ makes sense even if $tp \not\in r$). After line 08, is executed while the algorithm is processing $r[tp]$, $\Gamma_t$ will contain the (multi)set of all tp-cases from the tuples in $r[tp]$. Since $\gamma_i$ had been added to $tp$ in line 11., it means that there exists such a tp-case $\gamma^t = \langle C', D', L', U', \delta' \rangle \in Gamma^1$ that $sol(C_j) \subseteq sol(C')$. $D_j = D', L_j = L', U_j = U'$ and $\delta_j = \delta'$. Also this means that no other tp-case in $\Gamma_t$ refers to any timepoints in $sol(C_j)$, in particular, no other tp-case in $\Gamma_t$ refers to timepoint $t$. Clearly then $ANN(r)[d, t] = ANN(r[tp])[d, t]$ will contain one and only one annotated tuple: $at^t = (d, t, \delta(D', t) \cdot L', \delta(D', t) \cdot U')$. But from the equalities established above, $at^t = (d, t, \delta_j(D_j, t) \cdot L_j, \delta_j(D_j, t) \cdot U_j) = (d, t, L, U) = at$. Now we notice that $\chi([\{L, U\}]) = [L, U] = \{\}$ as it is required by the definition of $\chi$-compaition, since $\chi$ as a combination function satisfies identity postulate.

2. $\gamma_i$ had been added to $tp$ in line 22. In this case in the set $\Gamma_t$ built in line 08. for the equivalence class of $tp$ in $r$ there will be more than one tp-case which refers to $t$. From the analysis of the algorithm above we conclude that in this case $t \in sol(C_j)$ after $C_j$ had been computed in line 13. Let us consider the iteration of the foreach loop 15–22. which processes $t$. In line 17, $\Gamma_t$ gets assigned the value of the set of all tp-cases $\gamma_i = \langle C_i, D_i, L_i, U_i, \delta_i \rangle$ that $t \in sol(C_i)$. foreach loop in lines 18–20 builds the set $X = \{[L_i^1, U_i^1], \ldots, [L_i^s, U_i^s] \}$ where $[L_i^1, U_i^1] = [\delta_i(D_i, t) \cdot L_i, \delta_i(D_i, t) \cdot U_i]$ are the probabilistic intervals assigned to $d, t$ by each tp-case $\gamma_i$.

In line 21, $[L, U] = \chi(X)$ is computed. Now, all we have to prove is that $X = \{[L_1^{d, t}, U_1^{d, t}], \ldots, [L_k^{d, t}, U_k^{d, t}] \}$ such that $ANN(r)[d, t] = \{at_1^{d, t}, \ldots, at_k^{d, t}\}$ and $at_i^{d, t} = (d, t, L_i^{d, t}, U_i^{d, t})$. Let $\Gamma_t = \{\gamma_1^t, \ldots, \gamma_s^t\}$. Since $\Gamma_t$ contains all tp-cases of $r[tp]$ (and hence, all tp-cases of $r$) which refer to timepoint $t$ for the data $d$, $ANN(r)[d, t] = ANN(r[tp])[d, t]$ will be equal to the set: $\{at_1^t, \ldots, at_k^t\}$, where $at_i^t = (d, t, \delta_i(D_i, t) \cdot L_i, \delta_i(D_i, t) \cdot U_i) = (d, t, L_i^*, U_i^*), 1 \leq i \leq s$. But then we also know that $X = \{[L_1^*, U_1^*], \ldots, [L_k^*, U_k^*] \}$. This proves the theorem. □

Proof of Lemma 2.
Similar to the proof of Proposition 5. □

Proof of Theorem 8.
Similar to the proof of Theorem 7. □

Proof of Theorem 9.
By definition 6.7 $ANN(\kappa_\chi(r)) = \{at = (d, t, L, U) \vert [L, U] = \chi(\{[L_1^{d, t}, U_1^{d, t}], \ldots, [L_k^{d, t}, U_k^{d, t}]\})\}$, where $(\forall 1 \leq i \leq k)at_i^{d, t} = (d, t, L_i^{d, t}, U_i^{d, t})$ and $\{at_1^{d, t}, \ldots, at_k^{d, t}\} = ANN(r)[d, t]$. But by definition 5.2 $\kappa_\chi(ANN(r))$ is equal to the same expression. Therefore, $\kappa_\chi(ANN(r)) = \kappa_\chi(ANN(r))$. □

Proof of Theorem 10.
By the definition of a p-strategy based compaction of TP-relations $ANN(\kappa_p(r)) = \{at = (d, t, L, U) \vert [L, U] = \}$
where \((\forall 1 \leq i \leq k) a_i \equiv (d, t, L_i, r_i)\); \(\{a_1, \ldots, a_k\} = ANN(r)[d, t]\) and \(* = \oplus\) if \(\rho\) is a disjunctive p-strategy and \(* = \ominus\) if \(\rho\) is a conjunctive p-strategy.

But by the definition of a p-strategy based compaction of annotated relations \(\kappa_\rho(ANN(r))\) is equal to the same expression. Therefore, \(ANN(\kappa_\rho(r)) = \kappa_\rho(ANN(r))\).

**Proof of Theorem 11.**

By definition \(r \cap r' = \kappa_\rho(r \cap r')\). Our proof consists of three parts. show that:

1. **Claim 1:** \(\kappa_\rho(r \cap r') \subseteq \kappa_\rho(r) \cap \kappa_\rho(r')\).

Let \(at'' = (d, t, L, U) \in \kappa_\rho(r \cap r')\). We show that \(at'' \in \kappa_\rho(r) \cap \kappa_\rho(r')\). By definition of \(\kappa_\rho\), \(at'' \in \kappa_\rho(r \cap r')\) implies that there exists a tp-tuple \(t p'' \in r \cap r'\) such that:

\[
\begin{align*}
\gamma'' &= \gamma_1'' \land \cdots \land \gamma_n'' \\
\gamma' &= \gamma_1' \land \cdots \land \gamma_n' \\
\gamma_i &= \{C_i, D_i, L_i, U_i, \tilde{\epsilon}_i\} \\
\gamma'_i &= \{C_i', D_i', L_i', U_i', \tilde{\epsilon}'_i\}
\end{align*}
\]

As \(t p'' \in r \cap r'\), by definition of intersection of two tp-relations, there exist tp-tuples \(t p = (d, \gamma) \in r\) and \(t p' = (d, \gamma') \in r'\) where:

\[
\begin{align*}
\gamma &= \gamma_1, \ldots, \gamma_n \\
\gamma' &= \gamma_1', \ldots, \gamma_n' \\
\gamma_i &= \{C_i, D_i, L_i, U_i, \tilde{\epsilon}_i\} \\
\gamma'_i &= \{C_i', D_i', L_i', U_i', \tilde{\epsilon}'_i\}
\end{align*}
\]

In addition, there exist integers \(i_1 \in \{1, \ldots, n\}\) and \(i_2 \in \{1, \ldots, n'\}\) such that either

\[
\begin{align*}
C_{i_1}'' &= (C_{i_1} \land C_{i_2}) \text{ and} \\
D_{i_1}'' &= D_{i_1} \text{ and} \\
\tilde{\epsilon}_{i_1}'' &= \tilde{\epsilon}_{i_1} \text{ and} \\
[L_{i_1}'', U_{i_1}''] &= [L_{i_1}, U_{i_1}].
\end{align*}
\]

or

\[
\begin{align*}
C_{i_2}'' &= (C_{i_1} \land C_{i_2}') \text{ and} \\
D_{i_2}'' &= D_{i_2} \text{ and} \\
\tilde{\epsilon}_{i_2}'' &= \tilde{\epsilon}_{i_2} \text{ and} \\
[L_{i_2}'', U_{i_2}''] &= [L_{i_2}, U_{i_2}].
\end{align*}
\]

Without loss of generality, assume that the first case holds (the reasoning for the other case is symmetric). As \(t \in sol(C_{i_1})\) and \(t \in sol(C_{i_2}')\), by definition of \(\kappa_\rho\), we may say that \(\kappa_\rho(t) \ni at = (d, t, L_1, U_1)\) and \(\kappa_\rho(t) \ni at'' = (d, t, L_2, U_2)\) where \([L_1, U_1] = [\hat{\delta}(D_{i_1}, t) \cdot L_{i_1}, \hat{\delta}(D_{i_1}, t) \cdot U_{i_1}]\) and \([L_2, U_2] = [\hat{\delta}(D_{i_2}, t) \cdot L_{i_2}, \hat{\delta}(D_{i_2}, t) \cdot U_{i_2}]\). Therefore, by definition of intersection of two annotated relations, we conclude that \(\{at, at''\} \subseteq \kappa_\rho(r) \cap \kappa_\rho(r')\). It follows that \(at'' = (d, t, L, U) = at = (d, t, L_1, U_1)\), because \([L, U] = [L_{i_1}, U_{i_1}]\).
2. **Claim 2**: \( \text{ANN}(r \cap r') \supseteq \text{ANN}(r) \cap \text{ANN}(r'). \) Suppose \( at = (d,t,L,U) \in \text{ANN}(r) \cap \text{ANN}(r'). \) Hence, by definition of intersection there exists an annotated tuple \( at' = (d',t',L',U') \in \text{ANN}(r) \cap \text{ANN}(r'), \) such that

- either \( at \in \text{ANN}(r); at' \in \text{ANN}(r') \)
- or \( at' \in \text{ANN}(r); at \in \text{ANN}(r') \)

**Without loss of generality** we will assume that \( at \in \text{ANN}(r); at' \in \text{ANN}(r') \) (the other case is symmetric).

As \( at \in \text{ANN}(r) \) we conclude by definition of \( \text{ANN} \) that there exists a tp-tuple \( tp = (d, \gamma) \in r \) such that

\[
\forall 1 \leq i \leq n(\gamma_i = \langle C_i, D_i, L_i; U_i, \delta_i \rangle)
\]

and \( \exists i \in \{1, \ldots, n\} \) \( t \in \text{sol}(C_i) \land [L,U] = \{\delta_i(D_i,t)L_i; \delta_i(D_i,t)U_i\} \).

Now from \( at' \in \text{ANN}(r') \), we know that there exists a tp-tuple \( tp' = (d', \gamma') \in r' \) such that \( \langle tp, tp' \rangle \in m \) and

\[
\gamma' = \gamma'_1, \ldots, \gamma'_k
\]

\[
\forall 1 \leq i \leq n'(\gamma'_i = \langle C'_i, D'_i, L'_i; U'_i, \delta'_i \rangle)
\]

and \( \exists j \in \{1, \ldots, n'\} \) \( t \in \text{sol}(C'_j) \Lambda [L',U'] = \{\delta'_j(D'_j,t)L'_j; \delta'_j(D'_j,t)U'_j\} \).

As \( tp \) and \( tp' \) are data identical, there exists such a tp-tuple \( tp'' = (d, \gamma'') \in r \cap r' \) that

\[
\gamma'' = \gamma''_1, \ldots, \gamma''_n
\]

and \( \exists k \leq n''(\gamma''_k = \langle C''_k, D''_k, L''_k, U''_k, \delta''_k \rangle \land C''_k = C_k \land C'_j \land D''_k = D_i \land [L''_k, U''_k] = [L_i, U_i] \land \delta''_k = \delta_k \).

As \( t \in \text{sol}(C_i) \) and \( t \in \text{sol}(C'_j) \), \( t \in \text{sol}(C_i \land C'_j) = C''_k \). Therefore, \( \text{ANN}(t'') \) will contain a tuple \( at'' = (d,t,L'',U'') \) where

\[
[L'',U''] = [\delta''_k(D''_k,t) \cdot L''_k; \delta''_k(D''_k,t) \cdot U''_k] = [\delta_i(D_i,t) \cdot L_i; \delta_i(D_i,t) \cdot U_i].
\]

We can see that \( [L'',U''] = [L, U] \) and therefore, \( at'' = at \) and therefore \( at \in \text{ANN}(tp'') \), from which it follows that \( at \in \text{ANN}(r \cap r') \).

3. **Claim 3**: \( \text{ANN}(r \cap \chi) = \text{ANN}(r) \cap \text{ANN}(r') \).

By definition of the intersection of two TP relations \( \text{ANN}(r \cap \chi) \equiv \text{ANN}(\kappa_\chi(r \cap r')) \). As \( \langle \kappa_\chi, \chi \rangle \) is a compatible pair, we have \( \text{ANN}(\kappa_\chi(r \cap r')) = \kappa_\chi(\text{ANN}(r \cap r')) = \kappa_\chi(\text{ANN}(r) \cap \text{ANN}(r')) = \text{ANN}(r) \cap \text{ANN}(r') \). \( \square \)

**Proof of Theorem 12.**

Let \( r \) and \( r' \) be two tp-relations and \( \chi \) be a combination function. By definition of union \( r \cup \chi = \kappa_\chi (r \cup r') \) and \( \text{ANN}(r) \cup \chi \text{ANN}(r') = \kappa_\chi(\text{ANN}(r) \cup \text{ANN}(r')) \). Since \( \kappa, h, i(r) \) and \( \kappa_\chi(\text{ANN}(r)) \) is a compatible pair of compaction operations, it is sufficient to prove that \( \text{ANN}(r \cup r') = \text{ANN}(r) \cup \text{ANN}(r') \).

- \( \text{ANN}(r \cup r') \subseteq \text{ANN}(r) \cup \text{ANN}(r') \). Let \( at \in \text{ANN}(r \cup r') \). Then there exists a tp-tuple \( tp \in r \cup r' \), such that \( at \in \text{ANN}(tp) \). By definition of multiset union of two tp-relations, either \( tp \in r \) or \( tp \in r' \). If the former holds, \( at \in \text{ANN}(r) \) and if the latter holds, \( at \in \text{ANN}(r') \). In either case, \( at \in \text{ANN}(r) \cup \text{ANN}(r') \) as \( \text{ANN}(r) \cup \text{ANN}(r') \) consists of all annotated tuples in \( \text{ANN}(r) \) and all annotated tuples in \( \text{ANN}(r') \).

- \( \text{ANN}(r \cup r') \supseteq \text{ANN}(r) \cup \text{ANN}(r') \). Let \( at \in \text{ANN}(r) \cup \text{ANN}(r') \). Then, either \( at \in \text{ANN}(r) \) or \( at \in \text{ANN}(r') \). Assume the former is true (the other case is symmetric). Then, by def. of annotation operation, there exists a tp-tuple \( tp \in r \), such that \( at \in \text{ANN}(tp) \). But, since by definition of multiset union of two tp-relations \( tp \in r \cup r' \), we also get \( at \in \text{ANN}(r \cup r') \). \( \square \)
Proof of Theorem 13.
Let $C_1$ and $C_2$ be two atomic selection constraints. We will consider a number of cases, each for each pair of constraint types.

- If both $C_1$ and $C_2$ are data constraints, then the statement of the theorem follows from the similar result in relational algebra.
- If both $C_1$ and $C_2$ are temporal constraints, the statement of the theorem will be true because of commutativity of the conjunction of boolean temporal constraints.
- If both $C_1$ and $C_2$ are probabilistic constraints, the statement of the theorem will be true because of commutativity of the conjunction of boolean probabilistic constraints.
- Let $C_1$ be a data constraint and $C_2$ be a temporal constraint.

First we prove $\sigma_{C_1}(\sigma_{C_2}(r)) \subseteq \sigma_{\sigma_{C_1}(\sigma_{C_2}(r))}$. Let $t_p \in \sigma_{C_1}(\sigma_{C_2}(r))$. Then, (a) $t_p$ satisfies the data constraint $C_1$, and (b) $t_p \in \sigma_{C_2}(r)$. By definition of selection on atomic temporal constraint, there exists a tp-tuple $t_{p'} \in r$ such that

$$t_p = (d, \gamma), \gamma = \gamma_1, \ldots, \gamma_k, \gamma_i = \langle C_i, D_i, L_i, U_i, b_i \rangle, 1 \leq i \leq k;$$

$$t_{p'} = (d, \gamma'), \gamma' = \gamma'_1, \ldots, \gamma'_m, \gamma'_i = \langle C'_i, D_i, L_i, U_i, b_i \rangle, 1 \leq i \leq m$$

1. $m \geq k$;
2. and there exists a mapping $f : \{1, \ldots, k\} \rightarrow \{1, \ldots, m\}$ such that $f(i) = f(j)$ iff $i = j$ and $C_i = C'_{f(i)} \wedge C_2$.
3. If for some $1 \leq l \leq m$, $C'_l \wedge C_2$ is consistent, then there exists such number $1 \leq j \leq k$, that $f(j) = l$.

Since $t_p$ and $t_{p'}$ are data-identical, $t_{p'} \in \sigma_{C_1}(r)$, since $t_{p'}$ must also satisfy $C_1$. But then, by the definition of selection on temporal constraint $\sigma_{\sigma_{C_1}(\sigma_{C_2}(r))}$ will contain the tuple $t_{p''}$ defined as follows:

$$t_{p''} = (d, \gamma''), \gamma'' = \gamma''_1, \ldots, \gamma''_n, \gamma''_i = \langle C''_i, D_i, L_i, U_i, b_i \rangle, 1 \leq i \leq n$$

and for each $1 \leq j \leq m$ such that $C'_j \wedge C_2$ is consistent, there exists a unique $1 \leq i \leq n$ such that $C''_i = C'_j \wedge C_2$.

But the latter description is equivalent to the description of $t_p$, i.e. $t_{p''} = t_p$, i.e. $t_p \in \sigma_{C_2}(\sigma_{C_1}(r))$.

Now we prove that $\sigma_{C_1}(\sigma_{C_2}(r)) \supseteq \sigma_{\sigma_{C_1}(\sigma_{C_2}(r))}$. Let $t_p \in \sigma_{\sigma_{C_1}(\sigma_{C_2}(r))}$. This means that $(\sigma_{C_1}(r))$ contains a tuple $t_{p'}$ such that $t_p = (d, \gamma), \gamma = \gamma_1, \ldots, \gamma_k, \gamma_i = \langle C_i, D_i, L_i, U_i, b_i \rangle, 1 \leq i \leq k;$

$$t_{p'} = (d, \gamma'), \gamma' = \gamma'_1, \ldots, \gamma'_m, \gamma'_i = \langle C'_i, D_i, L_i, U_i, b_i \rangle, 1 \leq i \leq m$$

1. $m \geq k$;
2. and there exists a mapping $f : \{1, \ldots, k\} \rightarrow \{1, \ldots, m\}$ such that $f(i) = f(j)$ iff $i = j$ and $C_i = C'_{f(i)} \wedge C_2$.
3. If for some $1 \leq l \leq m$, $C'_l \wedge C_2$ is consistent, then there exists such number $1 \leq j \leq k$, that $f(j) = l$.

Since $t_{p'} \in \sigma_{C_1}(r)$ and $C_1(r), t_{p'} \in r$. Then $\sigma_{C_2}(r)$ will contain a tuple $t_{p''}$ such that $t_{p''} = (d, \gamma''),$

$$\gamma'' = \gamma''_1, \ldots, \gamma''_n, \gamma''_i = \langle C''_i, D_i, L_i, U_i, b_i \rangle, 1 \leq i \leq n$$

and for each $1 \leq j \leq m$ such that $C'_j \wedge C_2$ is consistent, there exists a unique $1 \leq i \leq n$ such that $C''_i = C'_j \wedge C_2$. Clearly, $t_p = t_{p''}$. Since $t_{p''}$ and $t_{p'}$ are data-identical, $t_{p''} \in \sigma_{C_1}(\sigma_{C_2}(r))$, i.e., $t_p \in \sigma_{C_1}(\sigma_{C_2}(r))$.

The proof for the case when $C_1$ is a temporal constraint and $\sigma_{C_2}$ is a data constraint is symmetric.
• $C_1$ is a data constraint and $\sigma_{C_2}$ is a probabilistic constraint.

First we prove $\sigma_{C_1}(\sigma_{C_2}(r)) \subseteq \sigma_{C_2}(\sigma_{C_1}(r))$. Let $tp \in \sigma_{C_1}(\sigma_{C_2}(r))$. Then, (a) $tp$ satisfies the data constraint $C_1$, and (b) $tp \in \sigma_{C_2}(r)$. By definition of selection on atomic probabilistic constraint, there exists a tp-tuple $tp' \in r$ such that $tp = (d, \gamma), \gamma = \gamma_1, \ldots, \gamma_k, \gamma_i = \langle C_i, D_i, L_i, U_i, b_i \rangle, 1 \leq i \leq k$; $tp' = (d, \gamma'), \gamma' = \gamma_1', \ldots, \gamma_m', \gamma_i' = \langle C_i', D_i, L_i, U_i, b_i \rangle, 1 \leq i \leq m$ and

1. $m \geq k$;
2. and there exists a mapping $f : \{1, \ldots, k\} \rightarrow \{1, \ldots, m\}$ such that $f(i) = f(j)$ iff $i = j$ and $C_1 = TP - filter(\gamma', \gamma''_i, C_2)$.
3. If for some $1 \leq l \leq m$, $sol(TP - filter(\gamma'_l, C_2)) \neq \emptyset$ then there exists such number $1 \leq j \leq k$, that $f(j) = l$.

Then by definition of selection on atomic data constraint, $\sigma_{C_1}(\sigma_{C_2}(r))$ will contain $tp'$ as $tp$ and $tp'$ are data-identical and $tp$ satisfies $C_1$. Therefore, $\sigma_{C_1}(\sigma_{C_2}(r))$ will contain a tp-tuple $tp'' = (d, \gamma'')$ constructed as follows: for each $\gamma'_i \in \gamma'$ such that $sol(TP - filter(\gamma'_i, C_2)) \neq \emptyset$, there will be a case $\langle TP - filter(\gamma'_i, C_2), D_i, L_i, U_i, b_i \rangle$ in $\gamma''$ and there will be no other cases in $\gamma''$. But it is clear that in this case $tp'' = tp$ and therefore $tp \in \sigma_{C_2}(\sigma_{C_1}(r))$.

To prove that $\sigma_{C_1}(\sigma_{C_2}(r)) \supseteq \sigma_{C_2}(\sigma_{C_1}(r))$ now let $tp \in \sigma_{C_2}(\sigma_{C_1}(r))$.

In this case, $\sigma_{C_1}(r)$ will contain a tp-tuple $tp'$ such that $tp = (d, \gamma), \gamma = \gamma_1, \ldots, \gamma_k, \gamma_i = \langle C_i, D_i, L_i, U_i, b_i \rangle, 1 \leq i \leq k$; $tp' = (d, \gamma'), \gamma' = \gamma_1', \ldots, \gamma_m', \gamma_i' = \langle C_i, D_i, L_i, U_i, b_i \rangle, 1 \leq i \leq m$ and

1. $m \geq k$;
2. and there exists a mapping $f : \{1, \ldots, k\} \rightarrow \{1, \ldots, m\}$ such that $f(i) = f(j)$ iff $i = j$ and $C_1 = TP - filter(\gamma', \gamma''_i, C_2)$.
3. If for some $1 \leq l \leq m$, $sol(TP - filter(\gamma'_l, C_2)) \neq \emptyset$ then there exists such number $1 \leq j \leq k$, that $f(j) = l$.

Then by definition of selection on atomic data constraint, $tp'$ must satisfy $C_1(r)$ and therefore $tp' \in r$. But then $C_2(r)$ will contain $tp$ (see the construction of $tp'$ above to see why this is true). And since $tp$ and $tp'$ are data-identical, $tp$ will satisfy $C_1(r)$ and therefore $tp \in \sigma_{C_1}(\sigma_{C_2}(r))$.

The proof for the case when $C_1$ is a probabilistic constraint and $C_2$ is a data constraint is symmetric.

• $C_1$ is a temporal constraint and $C_2$ is a probabilistic constraint.

First we prove $\sigma_{C_1}(\sigma_{C_2}(r)) \subseteq \sigma_{C_2}(\sigma_{C_1}(r))$. Let $tp \in \sigma_{C_1}(\sigma_{C_2}(r))$. Then, by definition of a selection on an atomic temporal condition, $\sigma_{C_1}(r)$ will contain a tuple $tp'$ such that $tp = (d, \gamma), \gamma = \gamma_1, \ldots, \gamma_k, \gamma_i = \langle C_i, D_i, L_i, U_i, b_i \rangle, 1 \leq i \leq k$; $tp' = (d, \gamma'), \gamma' = \gamma_1', \ldots, \gamma_m', \gamma_i' = \langle C_i, D_i, L_i, U_i, b_i \rangle, 1 \leq i \leq m$ and

1. $m \geq k$;
2. and there exists a mapping $f : \{1, \ldots, k\} \rightarrow \{1, \ldots, m\}$ such that $f(i) = f(j)$ iff $i = j$ and $C_1 = C_{f(i)} \land C_2$.
3. If for some $1 \leq l \leq m$, $C_{f(i)} \land C_2$ is consistent, then there exists such number $1 \leq j \leq k$, that $f(j) = l$. 

76
Since \( tp' \in \sigma_{C_1}(r) \), by definition of selection on atomic probabilistic condition there exists a tuple \( tp'' \in r \) such that
\[
\begin{align*}
theta_{tp'} &= (d, \gamma'), \gamma'' = \gamma_1'' \rightarrow \ldots \rightarrow \gamma_n'', \gamma''_i = \langle C_i'', D_i, L_i, U_i, b_i \rangle, 1 \leq i \leq n
\end{align*}
\]
1. \( n \geq m \);
2. and there exists a mapping \( g : \{1, \ldots, m\} \rightarrow \{1, \ldots, n\} \) such that \( g(i) = g(j) \iff i = j \) and \( C_i' = TP - \text{filter}(\gamma''_1, C_2) \).
3. If for some \( 1 \leq l \leq n \), \( \text{sol}(TP - \text{filter}(\gamma''_g, C_2)) = \emptyset \) then there exists such number \( 1 \leq j \leq k \), that \( g(j) = l \).

From the above we obtain that \( C_i' = TP - \text{filter}(\gamma(g(f(i)), C_2), C_1) \).

Since \( tp'' \in r \), we know that \( \sigma_{C_2}(r) \) will contain a tp-tuple \( tp''' = (d, \gamma''') \) such that \( \gamma''' = \gamma''_1''' \rightarrow \ldots \rightarrow \gamma''_r''' \), \( \gamma''_i''' = \langle C_i''''', D_i, L_i, U_i, b_i \rangle, 1 \leq i \leq r \) and for each \( 1 \leq j \leq n \) such that \( C_j'''' \cap C_1 \) is consistent, there exists a unique \( 1 \leq i \leq r \) such that \( C_i''' = C_j'''' \cap C_2 \).

Finally, \( \sigma_{C_2}(r) \) will contain a tp-tuple \( tp^* = (d, \gamma^*) \) such that for each \( \gamma''_i \in \gamma'' \) such that \( \text{sol}(TP - \text{filter}(\gamma''_i, C_2)) = \emptyset \), there will be a case \( (TP - \text{filter}(\gamma''_i, C_2), D_i, L_i, U_i, b_i) \) in \( \gamma^* \) and there will be no other cases in \( \gamma^* \).

Now we will show that \( tp \) indeed is equal to \( tp^* \). As we noticed, every case in \( tp \) had its \( C \) constraint have a form \( C_i' = TP - \text{filter}(\gamma''(f(i)), C_2), C_1 \).

Now, it is easy to notice that since every constraint in \( tp'' \) had a form \( C''_i = C''_1 \cap C_1 \), every constraint is \( tp^* \) will be of the form \( C''_i = TP - \text{filter}(\gamma''_i, C_2) = TP - \text{filter}(\gamma''_i, C_1, D''_i, L''_i, U''_i, b''_i, C_2). \)

Clearly, if for some \( 1 \leq j \leq n \), there was such \( 1 \leq i \leq k \) that \( C_i = TP - \text{filter}(\gamma''_i, C_2) \cap C_1 \) then \( TP - \text{filter}(\gamma''_i, C_1, D''_i, L''_i, U''_i, b''_i, C_2) \) will be the value of some \( C_i'' \) from \( \gamma^* \).

We now show that \( TP - \text{filter}(\gamma''_1, C_2) \cap C_1 = TP - \text{filter}((C''_1 \cap C_1, D''_1, L''_1, U''_1, b''_1, C_1)) \).

Let \( t \in \text{sol}(TP - \text{filter}(\gamma''_1, C_2) \cap C_1) \). In this case, \( t \in \text{sol}(C''_1) \) and \( \text{sol}(D''_1, t) \) satisfies \( C_2 \). Also, \( t \) satisfies \( C_1 \). But then, it satisfies \( C''_1 \cap C_1 \). Since this does not affect the probability estimate for \( t \), we obtain that \( t \in \text{sol}(TP - \text{filter}((C''_1 \cap C_1, D''_1, L''_1, U''_1, b''_1, C_1)) \).

Conversely, if \( t \in \text{sol}(TP - \text{filter}((C''_1 \cap C_1, D''_1, L''_1, U''_1, b''_1, C_1)) \) then (a) \( \text{sol}(D''_1, t) \) satisfies \( C_2 \) and (b) \( t \) satisfies \( C''_1 \cap C_1 \). Therefore \( t \) satisfies \( C''_1 \) and \( t \) satisfies \( C_1 \). Then clearly, \( t \in \text{sol}(TP - \text{filter}((C''_1, D''_1, L''_1, U''_1, b''_1, C_2)) \). Therefore, \( t \in \text{sol}(TP - \text{filter}((C''_1, D''_1, L''_1, U''_1, b''_1, C_2)) \cap \text{sol}(C_1) \), i.e. \( t \in \text{sol}(TP - \text{filter}(\gamma''_1, C_2) \cap C_1) \).

This proves the desired inclusion.

The proof that \( \sigma_{C_1}(r) = \sigma_{C_2}(r) \) is symmetric to the proof above. Similarly, the proof of the statement of the theorem for the case when \( C_1 \) is a probabilistic condition and \( C_2 \) is a temporal condition is symmetric to the proof above.

The list enumerates all possible pairs of types of constraints, therefore the theorem is proven.

\[ \square \]

**Proof of Theorem 14.**

We will prove this theorem for the case when \( C \) is an atomic predicate (constraint). Then, by theorem 13 this theorem will be true for arbitrary constraints as well.

So, assume \( C \) is atomic predicate. We have to consider three cases:

1. \( C \) is a predicate on the data part of the relational schema of \( r \).

   In this case we notice that \( \sigma_C(r) \subseteq r \), i.e. only tuples from \( r \) can be found in \( \sigma_C(r) \).
Let $at = (d, t, L, U) \in ANN(\sigma_C(r))$. By definition of annotation operation, there must be a tuple $tp = (d, \gamma) \in \sigma_C(r)$, such that $at \in ANN(tp)$. Since $C$ is a predicate on the data part, it must be the case that $d$ satisfies $C$.

But since $\sigma_C(r) \subseteq r$, we know that $tp \in r$, therefore, since $at \in ANN(tp)$, $at \in ANN(r)$. Finally, as $d$ satisfies $C$ we get $at \in \sigma_C(ANN(r))$.

To prove the inclusion the other way around, let us consider the tuple $at = (d, t, L, U) \in \sigma_C(ANN(r))$. Since $C$ is a predicate on the data part $d$ satisfies $C$.

By definition of selection on annotated tuples, $at \in ANN(r)$. But then, there must be a tp-tuple $tp = (d, \gamma) \in r$ such that $at \in ANN(tp)$. since $d$ satisfies $C$, $tp \in \sigma_C(r)$, and therefore, $at \in ANN(\sigma_C(r))$.

2. $C$ is a temporal predicate.

- $ANN(\sigma_C(r)) \subseteq \sigma_C(ANN(r))$
  
  Let $at = (d, t, L, U) \in ANN(\sigma_C(r))$. We show that $at \in \sigma_C(ANN(r))$.
  
  $at \in ANN(\sigma_C(r))$ implies that there exists a tp-tuple $tp = (d, \gamma) \in \sigma_C(r)$, such that $\gamma = \gamma_1, \ldots, \gamma_n$;
  
  $at \in ANN(tp)$ and
  
  $(\exists i \in \{1, \ldots, n\})(\gamma_i = \langle C_i, D_i, L_i, U_i, b_i \rangle) \land t \in sol(C_i) \land [L, U] = [b_i(D_i, t)L_i, b_i(D_i, t)U_i]$.
  
  Since $tp \in \sigma_C(r)$, by definition of selection on a TP-relation, there must exist a tp-tuple $tp' \in r$, such that:
  
  $tp' = (d, \gamma'); \gamma' = \gamma'_1, \ldots, \gamma'_k$ and
  
  $(\exists j \in \{1, \ldots, k\})(\gamma'_j = \langle C'_j, D'_j, L'_j, U'_j, b'_j \rangle)$ such that
  
  $\langle D'_j, L'_j, U'_j, b'_j \rangle = \langle D_i, L_i, U_i, b_i \rangle$ and
  
  $C'_j \cap C$.
  
  Since $t \in sol(C_i)$, $t \in sol(C'_j)$ and $t \in sol(C)$. The former means that $at \in ANN(tp')$, i.e., $at \in ANN(r)$. The latter means that by definition of selection on annotated relation, $at \in \sigma_C(ANN(r))$.

- $ANN(\sigma_C(r)) \supseteq \sigma_C(ANN(r))$
  
  Let $at = (d, t, L, U) \in \sigma_C(ANN(r))$. We show that $at \in ANN(\sigma_C(r))$.
  
  Since $at \in \sigma_C(ANN(r))$, by definition of selection on annotated relations
  
  (i) $at \in ANN(r)$
  
  (ii) $t \in sol(C)$
  
  Since $at \in ANN(r)$, there exists a tp-tuple $tp = (d, \gamma) \in r$, such that $\gamma = \gamma_1, \ldots, \gamma_n$;
  
  $at \in ANN(tp)$ and
  
  $(\exists i \in \{1, \ldots, n\})(\gamma_i = \langle C_i, D_i, L_i, U_i, b_i \rangle) \land t \in sol(C_i) \land [L, U] = [b_i(D_i, t)L_i, b_i(D_i, t)U_i]$.
  
  We know that $t \in sol(C_i)$ and $t \in sol(C)$, therefore $t \in sol(C_i \cap C)$. This means that there exists a tp-tuple $tp' \in \sigma_C(r)$ such that,
  
  $tp' = (d, \gamma'); \gamma' = \gamma'_1, \ldots, \gamma'_k$ and
  
  $(\exists j \in \{1, \ldots, k\})(\gamma'_j = \langle C'_j, D'_j, L'_j, U'_j, b'_j \rangle)$ such that
  
  $\langle D'_j, L'_j, U'_j, b'_j \rangle = \langle D_i, L_i, U_i, b_i \rangle$ and
  
  $C'_j \cap C$.
  
  $t \in sol(C'_j)$ and therefore, $at \in ANN(tp')$, i.e., $at \in ANN(\sigma_C(r))$.

3. $C$ is a probabilistic predicate.
We break the proof into two parts:

1. \( \text{ANN}(r - r') \subseteq \text{ANN}(r) - \text{ANN}(r') \)

   Let \( at = (d, t, L, U) \in \text{ANN}(r - r') \). We will show that \( at \in \text{ANN}(r) - \text{ANN}(r') \). As \( at \in \text{ANN}(r - r') \), by definition of \( \text{ANN} \), there exists such a tuple \( t' \in r - r' \) such that \( at \in \text{ANN}(t') \). Two cases have to be considered here:

   (a) \( t'' = (r - r') \) because \( t'' \in r \) and there is no tp-tuple in \( r' \) which is data identical to it. In this case, there is no annotated tuple in \( \text{ANN}(r') \) which is data identical to \( \text{ANN}(t'' \cap r') \) and hence, \( at \in \text{ANN}(r) - \text{ANN}(r') \).

   (b) Otherwise, \( t'' \) is in \( (r - r') \) because there is a tp-tuple \( t'' = (d, r') \) in \( r' \) and \( t'' \) is constructed from these two tp-tuples using the construction shown in Definition 6.17. Let \( (t, t') \) be any such pair of tp-tuples.

2. \( \text{ANN}(\sigma_{C}(r)) \subseteq \text{ANN}(\sigma_{C}(r)) \)

   Let \( at = (d, t, L, U) \in \text{ANN}(\sigma_{C}(r)) \). We show that \( at \in \text{ANN}(\sigma_{C}(r)) \).

   \( at \in \text{ANN}(\sigma_{C}(r)) \) implies that there exists a tp-tuple \( t' = (d, r') \in \sigma_{C}(r) \), such that \( r' \subseteq \bigcup \sigma_{C}(r) \).

   Since \( t' \in \sigma_{C}(r) \), by definition of selection on a TP-relation, there must exist a tp-tuple \( t'' \in r' \), such that:

   \[
   t'' = (d, \gamma') \quad \text{with} \quad \gamma' = \gamma_{1}, \ldots, \gamma_{n}
   \]

   Let us consider tuple \( t'' \in r' \), such that:

   \[
   t'' = (d, \gamma') \quad \text{with} \quad \gamma' = \gamma_{1}, \ldots, \gamma_{n}
   \]

   Since \( t'' \in \sigma_{C}(r) \), we know that \( t'' \in \text{sol}(\sigma_{C}(r)) \) and therefore, \( at \in \text{ANN}(\sigma_{C}(r)) \).

Proof of Theorem 15.

We break the proof into two parts:

1. \( \text{ANN}(r - r') \subseteq \text{ANN}(r) - \text{ANN}(r') \)

   Let \( at = (d, t, L, U) \in \text{ANN}(r - r') \). We will show that \( at \in \text{ANN}(r) - \text{ANN}(r') \). As \( at \in \text{ANN}(r - r') \), by definition of \( \text{ANN} \), there exists such a tuple \( t'' = (d, r'') \in r - r' \) such that \( at \in \text{ANN}(t'') \). Two cases have to be considered here:

   (a) \( t'' \) is in \( (r - r') \) because \( t'' \in r \) and there is no tp-tuple in \( r' \) which is data identical to it. In this case, there is no annotated tuple in \( \text{ANN}(r') \) which is data identical to \( \text{ANN}(t'' \cap r') \) and hence, \( at \in \text{ANN}(r) - \text{ANN}(r') \).

   (b) Otherwise, \( t'' \) is in \( (r - r') \) because there is a tp-tuple \( t'' = (d, r') \) in \( r' \) and \( t'' \) is constructed from these two tp-tuples using the construction shown in Definition 6.17. Let \( (t, t') \) be any such pair of tp-tuples.
As $at = (d, t, L, U) \in ANN(r \times r')$, it follows that there is a unique integer $i$ such that the constraint of the form $C_i \wedge \neg C'$ in Step (2) of Definition 6.17 is satisfied. This means that the time point $t$ is a solution of one of the $C$-constraints of $tp$ and none of the $C$-constraints of $tp'$. This holds for all tp-tuples in $r'$ that have an annotated tuple of the form $(d, t, \cdot, \cdot)$ in their annotated expansion. Hence, independently of how we choose a tuple $tp \in r'$, if $tp$ and $tp'$ are data-identical, then no tuple of the form $(d, t, \cdot, \cdot)$ can be in $ANN(tp) - ANN(tp')$. It follows that there can be no annotated tuple of the form $(d, t, \cdot, \cdot)$ in $ANN(r')$. If the $C$-constraint of $tp$ alluded to above is $\langle C_i, D_i, L_i, U_i, \delta_i \rangle$, then we know that $(d, t, L, U) \in ANN(r)$ where $L = \delta_i(D_i, t) \cdot L_i$ and $U = \delta_i(D_i, t) \cdot U_i$. Hence, $at \in ANN(r) - ANN(r')$.

2. $ANN(r) - ANN(r') \subseteq ANN(r - r')$.

Let $at = (d, t, L, U) \in ANN(r) - ANN(r')$. By definition, $at \in ANN(r)$ and there is no $at' = (d', t', L', U') \in ANN(r')$ such that $d = d'$ and $t = t'$. It follows by definition of $ANN$ that there exists $tp \in r$ such that $at \in ANN(tp)$ and there is no $tp' \in r'$ such that $ANN(tp')$ is of the form $(d, t, \cdot, \cdot)$. It follows immediately from the construction of $r - r'$ (Definition 6.17) that $at \in ANN(r - r')$.

Proof of Theorem 16.

The proof will consist of two parts

1. $ANN(r \times r) \subseteq ANN(r \times r')$.

Let $at'' = (d'', t'', L'', U'') \in ANN(r \times r')$. We will show that $at'' \in ANN(r \times r')$.

By the definition of Cartesian product of two annotated relations, there exists an annotated tuple $tp'' \in r \times r'$ such that $at'' \in ANN(tp'')$.

This means that $tp'' = (d'', t'')$, $\gamma'' = \gamma''_1, \ldots, \gamma''_m$, $\gamma''_i = \langle C''_i, D''_i, L''_i, U''_i, \delta_i \rangle$, $1 \leq i \leq n$ and there exists a number $1 \leq j \leq n$ such that $sol(C''_j) = \{t'' \}$ and $[L'', U''] = [L''_j, U''_j]$.

Since $tp'' \in r \times r'$, by the definition of Cartesian product of two tp-relations, there exists such a tp-tuple $tp \in r$ and a tp-tuple $tp' \in r'$ that:

$tp = (d, t)$

$tp' = (d', t')$

$\gamma = \gamma_1, \ldots, \gamma_k$

$\gamma_i = \langle C_i, D_i, L_i, U_i, \delta_i \rangle$, $1 \leq i \leq k$

$(\exists h \in \{1, \ldots, k\}) (t'' \in sol(C_h))$

$\gamma'_i = \langle C'_i, D'_i, L'_i, U'_i, \delta'_i \rangle$, $1 \leq i \leq m$

$(\exists l \in \{1, \ldots, m\}) (t'' \in sol(C'_l))$

and $[L''_j, U''_j] = [L'', U''] = [\delta_h(D_h, t''), L_h, \delta_h(D_h, t'') \cdot U_h] \otimes [\delta'_l(D'_l, t''), L'_l, \delta'_l(D'_l, t'') \cdot U'_l]$.

But in this case, by the definition of annotated relation, there will be an annotated tuple $at = (d, t'', \delta_h(D_h, t'') \cdot L_h, \delta_h(D_h, t'') \cdot U_h) \in ANN(tp) \subseteq ANN(r)$ and an annotated tuple $at' = (d', t'', \delta'_l(D'_l, t'') \cdot L'_l, \delta'_l(D'_l, t'') \cdot U'_l) \in ANN(tp') \subseteq ANN(r')$. Then by the definition of Cartesian product of annotated relations and since $[L'', U''] = [\delta_h(D_h, t'') \cdot L_h, \delta_h(D_h, t'') \cdot U_h] \otimes [\delta'_l(D'_l, t'') \cdot L'_l, \delta'_l(D'_l, t'') \cdot U'_l]$, $at'' = (d, d', t'', L'', U'') \in ANN(r \times r')$.

2. $ANN(r) \times r \subseteq ANN(r \times r')$.

Let $at'' = (d'', t'', L'', U'') \in ANN(r \times r')$. We will show that $at'' \in ANN(r \times r')$.

By the definition of Cartesian product of two annotated relations there exists an annotated tuple $at = (d, t, L, U) \in ANN(r)$ and an annotated tuple $at' = (d', t', L', U') \in ANN(r')$ such that $d'' = d, d'$ and $[L'', U''] = [L, U] \otimes [L', U']$. 

80
Since \( at \in ANN(r) \), there exists a tp-tuple \( tp \in r \) such that \( at \in ANN(tp) \), i.e., \( tp = (d, \gamma) \), \( \gamma = \gamma_1, \ldots, \gamma_k \), \( \gamma_i = \langle C_i, D_i, L_i, U_i, \delta_i \rangle \), \( 1 \leq i \leq k \) and \( (\exists h \in \{1, \ldots, k\})(t'' \in sol(C_h)) \) such that \([L, U] = [\delta_h(D_h, t''), L_h, \delta_h(D_h, t''), U_h] \).

Similarly, there exists a tp-tuple \( tp' \in r' \) such that \( at' \in ANN(tp') \), i.e., \( tp' = (d', \gamma') \), \( \gamma' = \gamma'_1, \ldots, \gamma'_m \), \( \gamma'_i = \langle C'_i, D'_i, L'_i, U'_i, \delta'_i \rangle \), \( 1 \leq i \leq m \) and \( (\exists l \in \{1, \ldots, m\})(t'' \in sol(C'_l)) \) and \([L', U'] = [\delta'_l(D'_l, t''), L'_l, \delta'_l(D'_l, t''), U'_l] \).

Since \( t'' \in C_h \) and \( t'' \in C'_l \), by definition of cartesian product of two tp-relations, \( r \times_\alpha r' \) will contain a tp-tuple \( tp'' = (d, d', \gamma, \gamma') \), \( \gamma'' = \gamma_1, \ldots, \gamma_n, \gamma'' = \langle C''_i, D''_i, L''_i, U''_i, \delta''_i \rangle \), \( 1 \leq i \leq n \) such that there exists a number \( 1 \leq j \leq n \) such that \( sol(C''_j) = \{t''\} \) and \([L'', U''] = [L, U] \otimes_\alpha [L', U'] = [L'', U''] \). But then \( ANN(r \times_\alpha r') \) will contain the tuple \( at'' = ((d, d'), t'', L'', U'') \).

### Proof of Theorem 17.

As usual the proof has two parts.

1. \( ANN(\pi_{\mathcal{F}, \alpha}(r)) \subseteq \pi_{\mathcal{F}, \alpha}(ANN(r)) \).

   Suppose \( at = (d, t, L, U) \in ANN(\pi_{\mathcal{F}, \alpha}(r)) \). By definition, there is a tp-tuple in \( \pi_{\mathcal{F}, \alpha}(r) \) of the form \( tp = (d, \gamma) \) in \( \pi_{\mathcal{F}, \alpha}(r) \) such that \( at \in ANN(tp) \). Let \( \gamma = \gamma_1, \ldots, \gamma_n \) and let \( \gamma_i = \langle C_i, D_i, L_i, U_i, \delta_i \rangle \) — hence, there is a unique integer \( 1 \leq j \leq n \) such that \( t \in sol(D_j) \), and \( L = \delta_j(D_j, t) \cdot L_j \) and \( U = \delta_j(D_j, t) \cdot U_j \). As TP-projection is a multiset operation, there is a unique tuple, \( tp^* \) in \( r \) such that \( tp.H = tp^*.H \parallel "A_1 : d.A_1, \ldots, A_n : d.A_n" \) and for all attributes in \( F \), \( tp^* \)'s attribute values and those of \( tp \) coincide. Hence, \( at \in \pi_{\mathcal{F}, \alpha}(ANN(tp^*)) \).

2. \( \pi_{\mathcal{F}, \alpha}(ANN(r)) \subseteq ANN(\pi_{\mathcal{F}, \alpha}(r)) \).

   Suppose \( at = (d, t, L, U) \in \pi_{\mathcal{F}, \alpha}(ANN(r)) \). Then there is an annotated tuple, \( at^* \in ANN(r) \) such that \( at = \pi_{\mathcal{F}, \alpha}(ANN(at^*)) \). But then there is a TP-tuple \( tp^* \) in \( r \) such that \( at^* \in tp^* \). Hence, \( at \in \pi_{\mathcal{F}, \alpha}(ANN(tp^*)) \).

### Proof of Theorem 18.

As join is a derived operation defined in terms of selection, projection and cartesian product operations, the result follows immediately from Theorems 14, 17 and 16.