Probabilistic Temporal Databases, II: Calculus and Query Processing*

Alex Dekhtyar† Fatma Özcan‡ Robert Ross § V.S. Subrahmanian¶

Abstract

There is a vast class of applications in which we know that a certain event occurred, but do not know exactly when it occurred. However, as studied by Dyreson and Snodgrass [7], there are many natural scenarios where probability distributions exist and quantify this uncertainty. Dekhtyar et. al. extended Dyreson and Snodgrass's work and defined an extension of the relational algebra to handle such data [5]. The first contribution of this paper is a declarative temporal probabilistic (TP for short) calculus which we show is equivalent in expressive power to the temporal probabilistic algebra of [5]. Our second major contribution is a set of equivalence and containment results for the TP-algebra. Our third contribution is the development of cost models that may be used to estimate the cost of TP-algebra operations. Our fourth contribution is an experimental evaluation of the accuracy of our cost models and the use of the equivalence results as rewrite rules for optimizing queries by using an implementation of TP-databases on top of ODBC.

1 Introduction

Dyreson and Snodgrass [7] pioneered the study of uncertainty in temporal databases where statements of the form "Event e occurred or will occur at some time point in the time interval $[t_1, t_2]$" are permitted. Such statements are common. For instance, a shipper like Federal Express may tell customers that their package will be delivered within 24 hours of dropoff. In such a case, if the smallest unit of time about which reasoning is performed is a minute, then there are (24×60) minutes at which the package may be possibly delivered. However, over this time-frame, there is a probability distribution reflecting the probability that the package will be delivered precisely t minutes after dropoff. Such a probability distribution may be skewed (e.g. the probability that the package is delivered within 3 hours is probably zero if the package has to make its way from Seattle to Boston). Dyreson and Snodgrass [7] developed this idea substantially to provide a framework for reasoning about temporal-probabilistic data. They also provided applications in a variety of other areas including carbon-dating of historical records and stock market analysis. For example, there are literally hundreds of programs that make stock market predictions — most prediction models

*This work was supported in part by the Army Research Lab under contract number DAAL0197K0135, the Army Research Office under grant number DAAD190010484, by DARPA/RL contract number F306029910552, and by the ARL CTA on Advanced Decision Architectures.

†Department of Computer Science, University of Kentucky, Lexington, KY. Email: dekhtyar@cs.uky.edu

‡Department of Computer Science, University of Maryland, College Park, MD 20742. Email: fatma@cs.umd.edu

§Department of Computer Science, University of Maryland, College Park, MD 20742. Email: robross@cs.umd.edu

¶Department of Computer Science, University of Maryland, College Park, MD 20742. Email: vs@cs.umd.edu

are uncertain and provide probabilistic outputs. In the same vein, statistical models that track performance of machines and machine parts on a factory floor yield probabilistic estimates of when the parts will need to be repaired and/or replaced. Decision making programs for such applications typically execute calls to the results of suc predictions.

Following on the important work of Dyreson and Snodgrass [7], work, Dekhtyar et. al. [5] developed a "temporal probabilistic algebra" (TPA) which eliminated many of the assumptions in the framework of [7].

We start this paper with a brief overview of TP-databases in Section 2. We then move on to our specific contributions.

1. First, in Section 3, we develop a temporal probabilistic calculus (TP-calculus for short). We show that this calculus, which is similar to the safe relational calculus, has the nice property that it is equivalent in expressive power to the TP-algebra.

2. Second, in Section 4, we develop a set of query equivalence results in the TP-algebra. These equivalences automatically yield a set of rewrite rules that a query optimizer for TP-databases might use.

3. Third, in Section 5, we develop mechanisms to estimate the cost of executing a TP-algebra query. Even though TP-algebra operations are implemented on top of a relational database, these operations are not mere relational algebra queries — implementing them on top of the relational algebra involves writing a C++ program that includes embedded SQL calls. Costing such programs builds on top of cost models of relational operators, but involves taking into account, the specific aspects of the programs themselves. Our cost models involve identifying a set of statistics to be stored about the data, as well as methods to compute such statistics for intermediate results obtained during query processing.

4. Fourth, in Section 6, we conduct a set of experiments to evaluate three things. Our first goal is to assess the accuracy of our cost model. Our second goal is to assess the effectiveness of our rewrite rules. The third goal is to assess the efficiency of a query optimizer based on this cost model and these rewrite rules. Experiments were run using our implementation of TP-databases which is built on top of ODBC (using Paradox as a back-end).

We compare our work with related work in Section 7 and finally conclude with directions for further research.

2 Preliminaries: TP-Databases Overview

In this section, we recapitulate the notions of a Temporal Probabilistic database (TP-database) and the Temporal Probabilistic Algebra (TPA). Full details may be found in [5].

2.1 Temporal-Probabilistic DB Model

To make statements about when certain events occurred, or when certain facts were/will be true, TP-databases use a calendar (cf. Snodgrass and Soo [32]) and temporal constraints.

Calendar: A *calendar* τ consists of an ordered list of *time units*. For example, (day, month, year) is a possible calendar. Each time unit has a finite domain associated with it. Given a calendar τ over time units $(tu_1, \ldots, tu_n)$, a *time point* is an expression of the form $(v_1, \ldots, v_n)$ where each v_i is in the domain of tu_i. For instance, w.r.t. the example calendar, $(25, 5, 2000)$ is a time point.

Temporal constraint: *Temporal constraints* are inductively defined. (i) If tu is a time unit, $\mathsf{op} \in \{<, \leq, =, \neq, \geq, >\}$, and $v \in dom(tu)$, then $(tu\ \mathsf{op}\ v)$ is a temporal constraint over τ. (ii) If t_1, t_2 are time points in τ and $t_1 \leq t_2$, then $(t_1 \sim t_2)$ is a temporal constraint over τ. (iii) If C_1 and C_2 are temporal constraints then so are $(C_1 \wedge C_2)$, $(C_1 \vee C_2)$ and $(\neg C_1)$.

The syntax $(t_1 \sim t_2)$ is a shorthand for the constraint $(t_1 \leq t \leq t_2)$. We abuse notation and write (t_1) instead of $(t_1 \sim t_1)$. *Solutions* of temporal constraints (denoted $\mathrm{sol}(C)$) are defined in the obvious way. For example, the constraint $((12{:}05 \sim 12{:}14) \vee (12{:}45 \sim 12{:}50))$ has as its solutions all time points between 12:05pm to 12:15pm, as well as all time points between 12:45pm to 12:50pm.

Probability Distribution Function (PDF): Let $\mathcal{S}_\tau$ be the set of all temporal constraints over calendar τ. Then the function $\wp : \mathcal{S}_\tau \times \tau \to [0,1]$ is a *PDF* if $(\forall D \in \mathcal{S}_\tau)(\forall t \notin \mathrm{sol}(D))(\wp(D,t) = 0)$. Furthermore, $\wp$ is a *restricted PDF* if $(\forall D \in \mathcal{S}_\tau)(\sum_{t \in \mathrm{sol}(D)} \wp(D,t) \leq 1)$.

This definition of a PDF is rich enough to capture almost all probability mass functions (e.g. uniform, geometric, binomial, Poisson, etc.) studied in classical probability theory [25]. Furthermore, probability density functions can be approximated by PDFs via a process of quantization.

TP-case: A *TP-case* over calendar τ is a 5-tuple $\langle C, D, L, U, \delta \rangle$ where (i) C and D are temporal constraints over τ, (ii) $\emptyset \subset \mathrm{sol}(C) \subseteq \mathrm{sol}(D)$, (iii) $0 \leq L \leq U \leq 1$, and (iv) δ is a restricted PDF.

For example, $\langle (12{:}05 \sim 12{:}14), (12{:}05 \sim 12{:}19), 0.5, 0.6, u \rangle$ is a TP-case where δ is a uniform distribution. Intuitively, C specifies the time points when an event is valid while D specifies the time points that are distributed by δ. Since $\mathrm{sol}(C) \subseteq \mathrm{sol}(D)$, it follows that δ assigns a probability interval to each time point $t \in \mathrm{sol}(C)$. Specifically, let $\Pr(\langle C, D, L, U, \delta \rangle, t)$ denote $\delta(D,t) \cdot [L, U]$.

Alternatively, δ could be defined as $(\wp_L, \wp_U)$ where $\wp_L$ is a restricted PDF, $\wp_U$ is a (unrestricted) PDF, and $(\forall t \in \mathrm{sol}(C))(\wp_L(D,t) \cdot L \leq \wp_U(D,t) \cdot U)$. This generalization is useful when the lower and upper probability bounds do not follow the same distribution. Here, $\Pr(\langle C, D, L, U, (\wp_L, \wp_U) \rangle, t)$ denotes the probability interval $[\wp_L(D,t) \cdot L, \wp_U(D,t) \cdot U]$. Although extending δ to a pair of PDFs is straightforward, we avoid this redefinition in order to maintain better compatibility with [5].

TP-tuple: A *TP-tuple* over relation scheme $R = (A_1, \ldots, A_k)$ and calendar τ is a pair (d, Γ) where d is a relational k-tuple over R and Γ is a nonempty *TP-case statement* over τ, i.e., Γ is a set of TP-cases over τ where $(\forall \gamma_i, \gamma_j \in \Gamma)((i \neq j) \to \mathrm{sol}(\gamma_i.C \wedge \gamma_j.C) = \emptyset)$.

In the following, suppose R is a relation scheme and τ is a calendar over $(tu_1, \ldots, tu_n)$.

TP-table: A *TP-table* over R, τ is a multiset of TP-tuples over R, τ.

TP-relation: A *TP-relation* r over R, τ is a TP-table over R, τ where R is a superkey for r.

TP-database: A *TP-database* over τ is a set of TP-tables over τ.

Annotation: The *annotation* of TP-relation r over R, τ produces a relation $\mathrm{ANN}(r)$ over relation scheme $(R, tu_1, \ldots, tu_n, L_t, U_t)$ where $dom(L_t) = dom(U_t) = [0, 1]$ and

$$\mathrm{ANN}(r) = \{(d, t, L_t, U_t) \mid (\exists\, \Gamma)\, (\exists\, \gamma \in \Gamma)\, ((d, \Gamma) \in r \,\wedge\, t \in \mathrm{sol}(\gamma.C) \,\wedge\, [L_t, U_t] = \Pr(\gamma, t))\}$$

Equivalence: TP-relations r and r' are *equivalent*, denoted $r \equiv r'$, iff $\mathrm{ANN}(r) = \mathrm{ANN}(r')$.

Example 1 (Base TP-relations) The following TP-relations are named TrainDep and BusArr:

TrainNo	From	To	C	D	L	U	δ
151	Baltimore	New York	(12:05 $\sim$ 12:14)	(12:05 $\sim$ 12:14)	0.5	0.6	u
			(12:15 $\sim$ 12:20)	(12:15 $\sim$ 12:20)	0.3	0.4	$g, 0.5$

BusNo	From	To	C	D	L	U	δ
23	Rockville	Baltimore	(12:12 $\sim$ 12:16)	(12:12 $\sim$ 12:16)	0.5	0.5	$b, 0.5$
			(12:17 $\sim$ 12:26)	(12:17 $\sim$ 12:26)	0.5	0.5	u

The second TP-case in TrainDep says that there is a 30-40% probability that train number 151 from Baltimore to New York will depart between 12:15 and 12:20. Furthermore, given any time point t in this interval, the probability that the train will depart at exactly time t is between $(0.3) \cdot \delta_{g,0.5}(D, t)$ and $(0.4) \cdot \delta_{g,0.5}(D, t)$ where $D = (12{:}15 \sim 12{:}20)$. Note that when $\delta = u$, $\delta = g, 0.5$, or $\delta = b, 0.5$, then the PDF is uniform (where $n = |\mathrm{sol}(D)|$), geometric (where $p = 0.5$), or binomial (where $n = |\mathrm{sol}(D)|$ and $p = 0.5$) respectively. $\diamond$

2.2 Temporal-Probabilistic Algebra (TPA)

In this section, we briefly overview some (but not all) of the TPA operators in [5].

Definition 1 (Selection condition) Suppose R is a relation scheme, τ is a calendar, and $\Theta \in \{<, \leq, =, \neq, \geq, >\}$. Then $\mathcal{C}$ is a *selection condition over* R, τ if it has one of the following forms:

- Data condition: $\mathcal{C} = A \;\Theta\; c$ where $A \in R$ and constant $c \in dom(A)$.
- Temporal condition: $\mathcal{C} = T$ where T is a temporal constraint over τ.
- Probabilistic condition: $\mathcal{C} = P \;\Theta\; p$ where $P \in \{L, U\}$ and probability $p \in [0, 1]$.
- Conjunction condition: $\mathcal{C} = (\mathcal{C}_1 \wedge \ldots \wedge \mathcal{C}_n)$ where $\mathcal{C}_i$ is a selection condition over R, τ. $\diamond$

We abuse notation and write $A \in R$ to indicate that A is an attribute of R. It should be clear from context whether we mean set membership or membership of an attribute in R's schema.

Definition 2 (TP-filter) A *TP-filter function* maps a TP-case $\langle C, D, L, U, \delta \rangle$ and a probabilistic condition $P \;\Theta\; p$ to a temporal constraint C'' where $\mathrm{sol}(C'') = \{t \in \mathrm{sol}(C) \mid (\delta(D, t) \cdot P) \;\Theta\; p\}$. $\diamond$

Various methods for implementing TP-filter functions are given in [5]. For example, a naive approach involves testing every time point $t \in \mathrm{sol}(C)$. Better algorithms exploit the properties of δ. For instance if $\delta = u$, then C'' can be determined by testing only one time point in $\mathrm{sol}(C)$.

Definition 3 (Selection on a TP-relation) Suppose r is a TP-relation over $R, \tau, \mathcal{C}$ is a selection condition over R, τ, and *flt* is a TP-filter function. Then the *selection of* $\mathcal{C}$ *on* r *using flt*, denoted $\sigma_{\mathcal{C}}^{flt}(r)$, produces TP-relation r'' over R, τ where the following constraints are satisfied:

- If $\mathcal{C} = A \; \Theta \; c$, then $r'' = \{(d, \Gamma) \in r \mid d.A \; \Theta \; c\}$.
- If $\mathcal{C} = T$, then $r'' = \{(d, \Gamma'') \mid \Gamma'' = \{\langle (C \wedge T), D, L, U, \delta \rangle \mid (\exists \, \Gamma) \, ((d, \Gamma) \in r \; \wedge$ $\langle C, D, L, U, \delta \rangle \in \Gamma \; \wedge \; \mathrm{sol}(C \wedge T) \neq \emptyset)\} \; \wedge \; \Gamma'' \neq \emptyset\}$.
- If $\mathcal{C} = P \; \Theta \; p$, then $r'' = \{(d, \Gamma'') \mid \Gamma'' = \{\langle flt(\gamma, \mathcal{C}), D, L, U, \delta \rangle \mid (\exists \, \Gamma) \, (\exists \, C) \, ((d, \Gamma) \in r \; \wedge$ $\gamma = \langle C, D, L, U, \delta \rangle \in \Gamma \; \wedge \; \mathrm{sol}(flt(\gamma, \mathcal{C})) \neq \emptyset)\} \; \wedge \; \Gamma'' \neq \emptyset\}$.
- If $\mathcal{C} = (\mathcal{C}_1 \wedge \mathcal{C}_2)$, then $r'' = \sigma_{\mathcal{C}_2}(\sigma_{\mathcal{C}_1}(r))$. ◇

Although the implementation chosen for TP-filter function *flt* affects the efficiency of the selection operation, notice that $\sigma_{\mathcal{C}}^{flt}(r) \equiv \sigma_{\mathcal{C}}^{flt'}(r)$ must hold for all TP-filter functions flt, flt'. Thus, we let $\sigma_{\mathcal{C}}(r)$ denote $\sigma_{\mathcal{C}}^{flt}(r)$ with any choice for *flt*.

Example 2 (Selection) Let $r_1 = \sigma_{((12{:}05 \sim 12{:}08) \vee (12{:}11 \sim 12{:}12) \vee (12{:}15 \sim 12{:}17))}(\mathsf{TrainDep})$ and let $r_2 = \sigma_{L \geq 0.05}(\mathsf{BusArr})$. Then the following are the TP-relations for r_1 and r_2:

TrainNo	From	To	C	D	L	U	δ
151	Baltimore	New York	((12:05 ∼ 12:08) ∨ (12:11 ∼ 12:12))	(12:05 ∼ 12:14)	0.5	0.6	u
			(12:15 ∼ 12:17)	(12:15 ∼ 12:20)	0.3	0.4	$g, 0.5$

BusNo	From	To	C	D	L	U	δ
23	Rockville	Baltimore	(12:13 ∼ 12:15)	(12:12 ∼ 12:16)	0.5	0.5	$b, 0.5$
			(12:17 ∼ 12:26)	(12:17 ∼ 12:26)	0.5	0.5	u

Notice that for both selections, we only need to change the values for the C attribute. ◇

Definition 4 (Attribute list) Suppose relation scheme $R = (A_1, \ldots, A_k)$ and P is the primary key for R. Then $\mathcal{F} = a_1, \ldots, a_n$ is an *attribute list over* R, P if the following constraints are satisfied: (i) $n \geq 1$, (ii) each a_i is an attribute of R, (iii) each attribute in P is an attribute in $\mathcal{F}$, and (iv) no attribute occurs more than once in $\mathcal{F}$. ◇

Definition 5 (Projection on a TP-relation) Suppose $R = (A_1, \ldots, A_k)$, r is a TP-relation over R, τ, P is the primary key for R, and $\mathcal{F} = a_1, \ldots, a_n$ is an attribute list over R, P. Then the *projection of* $\mathcal{F}$ *on* r, denoted $\pi_{\mathcal{F}}(r)$, produces TP-relation r'' over R'', τ where $R'' = (\mathcal{F})$ and

$$r'' = \{(d'', \Gamma) \mid (\exists \, d) \, (\forall i \in [1, n]) \, ((d, \Gamma) \in r \; \wedge \; d''.a_i = d.a_i)\}$$ ◇

Note that our projection operator is exactly like projection in the classical relational algebra except the fields in the primary key for r and the C, D, L, U, δ fields cannot be projected out.

Before presenting our definition of Cartesian product, we need to recall the notion of a *probabilistic conjunction strategy* introduced by Lakshmanan et. al. [20]. Intuitively, this is a function that returns the probability interval for a conjunction of two events given the probability intervals for the individual events. We shall also introduce the concept of a *Cartesian product function.* These functions are applied within the inner loop of a Cartesian product.

Definition 6 (PCS) A *probabilistic conjunction strategy* (PCS) is a binary function $\otimes$ from closed probability intervals to closed probability intervals where the following postulates are satisfied:

1. Bottomline: $([L_1, U_1] \otimes [L_2, U_2]) \leq [\min(L_1, L_2), \min(U_1, U_2)]$.
2. Ignorance: $([L_1, U_1] \otimes [L_2, U_2]) \subseteq [\max(0, L_1 + L_2 - 1), \min(U_1, U_2)]$.
3. Identity: $([L_1, U_1] \otimes [1, 1]) = [L_1, U_1]$.
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 $[L_2, U_2] \leq [L_3, U_3]$. $\diamond$

The following are some sample probabilistic conjunction strategies:

PCS name	Probability interval returned
Ignorance	$([L_1, U_1] \otimes_{ig} [L_2, U_2]) = [\max(0, L_1 + L_2 - 1), \min(U_1, U_2)]$
Positive correlation	$([L_1, U_1] \otimes_{pc} [L_2, U_2]) = [\min(L_1, L_2), \min(U_1, U_2)]$
Negative correlation	$([L_1, U_1] \otimes_{nc} [L_2, U_2]) = [\max(0, L_1 + L_2 - 1), \max(0, U_1 + U_2 - 1)]$
Independence	$([L_1, U_1] \otimes_{in} [L_2, U_2]) = [L_1 \cdot L_2, U_1 \cdot U_2]$

Definition 7 (CPF) A *Cartesian product function* (CPF) maps a TP-case γ, a PCS $\otimes$, and a TP-case γ' to a TP-case statement Γ'' where (i) $\mathrm{sol}(\bigvee_{\gamma'' \in \Gamma''}(\gamma''.C)) = \mathrm{sol}(\gamma.C \wedge \gamma'.C)$ and (ii) $(\forall \gamma'' \in \Gamma'')\,(\forall t \in \mathrm{sol}(\gamma''.C))\,(\Pr(\gamma'', t) = \Pr(\gamma, t) \otimes \Pr(\gamma', t))$. $\diamond$

A simple CPF example is cpf_s. This function is defined as

$$cpf_s(\gamma, \otimes, \gamma') = \{\langle (t), (t), L, U, u\rangle \mid t \in \mathrm{sol}(\gamma.C \wedge \gamma'.C) \wedge [L, U] = \Pr(\gamma, t) \otimes \Pr(\gamma', t)\}$$

If δ is a pair of PDFs, then another CPF example is $cpf_p(\gamma, \otimes, \gamma') = \{\langle C'', D'', L'', U'', (\wp''_L, \wp''_U)\rangle\}$ where $C'' = (\gamma.C \wedge \gamma'.C)$, $D'' = C''$, $L'' = \min(\gamma.L, \gamma'.L)$, $U'' = \min(\gamma.U, \gamma'.U)$, and $\wp''_L, \wp''_U$ are new PDFs such that for each $t \in \mathrm{sol}(D'')$, $[\wp''_L(D'', t), \wp''_U(D'', t)] = [\ell_t, u_t]$, $\ell_t = \frac{L_t}{L''}$ if $L'' \neq 0$ or $\ell_t = 0$ otherwise, $u_t = \frac{U_t}{U''}$ if $U'' \neq 0$ or $u_t = 0$ otherwise, and $[L_t, U_t] = \Pr(\gamma, t) \otimes \Pr(\gamma', t)$.

If δ is a single PDF, then a hybrid CPF example is cpf_h. Specifically, $cpf_h(\gamma, \otimes, \gamma')$ is defined as $\{\langle C'', D'', L'', L'' \cdot x, \wp''_L\rangle\}$ if $cpf_p(\gamma, \otimes, \gamma') = \{\langle C'', D'', L'', U'', (\wp''_L, \wp''_U)\rangle\}$ and $(\exists\, x \in [1, \frac{1}{L''}])$ $(\forall t \in \mathrm{sol}(D''))\,(\wp''_L(D'', t) = \wp''_U(D'', t) \cdot x)$. Otherwise, $cpf_h(\gamma, \otimes, \gamma')$ is the result of $cpf_s(\gamma, \otimes, \gamma')$.

Definition 8 (Cartesian product of two TP-relations) Suppose r is a TP-relation over R, τ, r' is a TP-relation over R', τ, α is a probabilistic conjunction strategy, cpf is a Cartesian product function, and $\{A \mid A \in R \wedge A \in R'\} = \emptyset$. Then the *Cartesian product of r and r' under α using cpf*, denoted $r \times_{\alpha}^{cpf} r'$, produces TP-relation r'' over R'', τ where $R'' = (R, R')$ and

$$r'' = \{(d'', \Gamma'') \mid (\exists\, (d, \Gamma) \in r)\,(\exists\, (d', \Gamma') \in r')\,(d'' = (d, d') \wedge \Gamma'' \neq \emptyset \wedge \Gamma'' = \textstyle\bigcup_{\gamma \in \Gamma, \gamma' \in \Gamma'}(cpf(\gamma, \otimes_\alpha, \gamma')))\}$$

$\diamond$

Since $r \times_{\alpha}^{cpf} r' \equiv r \times_{\alpha}^{cpf'} r'$ must hold, we let $r \times_{\alpha} r'$ denote $r \times_{\alpha}^{cpf} r'$ with any choice for *cpf*.

Definition 9 (Renaming function) Suppose $R = (A_1, \ldots, A_k)$. Then a *renaming function over* R is a function $\mathcal{R}$ that maps each attribute $A_i \in R$ to an attribute $\mathcal{R}(A_i) \notin \{C, D, L, U, \delta\}$ where $(\forall A_i, A_j \in R)\,(\mathcal{R}(A_i) = \mathcal{R}(A_j) \rightarrow A_i = A_j)$. ◇

Definition 10 (Renaming for a TP-relation) Suppose $R = (A_1, \ldots, A_k)$, r is a TP-relation over R, τ, and $\mathcal{R}$ is a renaming function over R. Then the *renaming of* r *under* $\mathcal{R}$, denoted $\rho_{\mathcal{R}}(r)$, produces TP-relation r'' over R'', τ where $R'' = (\mathcal{R}(A_1), \ldots, \mathcal{R}(A_k))$ and

$$r'' = \{(d'', \Gamma) \mid (\exists\, d)\,(\forall A \in R)\,((d, \Gamma) \in r \,\wedge\, d''.\mathcal{R}(A) = d.A)\}$$ ◇

Example 3 (Renaming and Cartesian product) Let $\mathcal{R}_1$ map TrainNo, From, To to TrainNo, TrainFrom, TrainTo, let $\mathcal{R}_2$ map BusNo, From, To to BusNo, BusFrom, BusTo, and let $r_3 = \rho_{\mathcal{R}_1}(r_1) \times_{in} \rho_{\mathcal{R}_2}(r_2)$. Then the following is a possible TP-relation for r_3:

TrainNo	TrainFrom	TrainTo	BusNo	BusFrom	BusTo	C	D	L	U	δ
151	Baltimore	New York	23	Rockville	Baltimore	(12:15)	(12:15)	0.01875	0.025	u
						(12:17)	(12:17)	0.001875	0.0025	u

Notice that renaming operators can only change the names of the data attributes. ◇

We are now ready to extend the classical definition of a natural join to TP-databases.

Definition 11 (Join of two TP-relations) Suppose $R = (A_1, \ldots, A_k)$, $R' = (A'_1, \ldots, A'_{k'})$, r is a TP-relation over R, τ, r' is a TP-relation over R', τ, α is a probabilistic conjunction strategy, *cpf* is a Cartesian product function, and *flt* is a TP-filter function. Then the *join of* r *and* r' *under* α *using cpf and flt*, denoted $r \bowtie_{\alpha}^{cpf,flt} r'$, produces TP-relation r'' over R'', τ where $R'' = (\mathcal{F})$, $r'' = \pi_{\mathcal{F}}(\sigma_{\mathcal{C}}^{flt}(r \times_{\alpha}^{cpf} \rho_{\mathcal{R}}(r')))$, and $\mathcal{R}, \mathcal{C}, \mathcal{F}$ have the following values:

- $\mathcal{R}$ is a renaming function over R' where
  $(\forall A \in R')\,((A \notin R \rightarrow \mathcal{R}(A) = A) \,\wedge\, (A \in R \rightarrow \mathcal{R}(A) \notin R))$.
  This requirement indicates that only the attributes in R' that occur in R get renamed.
  These attributes must be renamed to attribute names that are not present in R.
- $\mathcal{C}$ is the selection condition $(a_1 = \mathcal{R}(a_1) \wedge \ldots \wedge a_n = \mathcal{R}(a_n))$ where
  $\{a_1, \ldots, a_n\} = \{a \in R' \mid a \in R\}$.
  This is the natural join predicate.
- $\mathcal{F}$ is the attribute list $A_1, \ldots, A_k, \mathcal{R}(a_{n+1}), \ldots, \mathcal{R}(a_{k'})$ where
  $\{a_{n+1}, \ldots, a_{k'}\} = \{a \in R' \mid a \notin R\}$ and $a_{n+1}, \ldots, a_{k'}$ is a sublist of $A'_1, \ldots, A'_{k'}$. ◇

Since $r \bowtie_{\alpha}^{cpf,flt} r' \equiv r \bowtie_{\alpha}^{cpf',flt'} r'$ must hold, we let $r \bowtie_{\alpha} r'$ denote $r \bowtie_{\alpha}^{cpf,flt} r'$ with any choice for *cpf* and *flt*. Now, let $\mathcal{N}(r)$ denote $\sum_{(d,\Gamma)\in r}(|\Gamma|)$, i.e., the total number of TP-cases in r. It is easy to see that $\mathcal{N}(r \bowtie_{\alpha} r')$ can be huge. The *TP-compression* operation helps alleviate this problem.

Definition 12 (TP-compression) A *TP-compression function* Ξ is a function that maps TP-relation r over R, τ to a TP-relation r'' over R, τ where (i) $\mathcal{N}(r'') \leq \mathcal{N}(r)$ and (ii) there exists a bijection that maps each $(d, t, L_t, U_t) \in \text{ANN}(r)$ to a matching $(d, t, L_t, U_t) \in \text{ANN}(r'')$. $\diamond$

The following are some sample TP-compression functions:

TP-compression function name	Function properties
Identity (Ξ_{id})	$\Xi_{id}(r) = r$
Maximal compression (Ξ_{mc})	$(\forall\, \Xi)\,(\mathcal{N}(\Xi_{mc}(r)) \leq \mathcal{N}(\Xi(r)))$

Let $|r|$ denote the number of TP-tuples in r. Then it must be the case that $\mathcal{N}(r) \geq |r|$. Note that if δ can be a pair of PDFs, then $\mathcal{N}(\Xi_{mc}(r)) = |\Xi_{mc}(r)| = |r|$ for every TP-relation r.

Definition 13 (Intersection of two TP-tables) Suppose r and r' are TP-tables over R, τ. Then the *multiset intersection of* r *and* r', denoted $r \cap r'$, produces TP-table r'' over R, τ where

$$\begin{aligned} r'' = \{(d, \Gamma'') \mid (\exists\, \Gamma)\,(\exists\, \Gamma')\,((d, \Gamma) \in r \,\wedge\, (d, \Gamma') \in r' \,\wedge\, \Gamma'' \neq \emptyset \,\wedge \\ (\Gamma'' = \{\langle (C \wedge \gamma'.C), D, L, U, \delta\rangle \mid \langle C, D, L, U, \delta\rangle \in \Gamma \,\wedge\, \gamma' \in \Gamma' \,\wedge\, \text{sol}(C \wedge \gamma'.C) \neq \emptyset\} \,\vee \\ \Gamma'' = \{\langle (\gamma.C \wedge C), D, L, U, \delta\rangle \mid \gamma \in \Gamma \,\wedge\, \langle C, D, L, U, \delta\rangle \in \Gamma' \,\wedge\, \text{sol}(\gamma.C \wedge C) \neq \emptyset\}))\} \end{aligned}$$ $\diamond$

Definition 14 (Union of two TP-tables) Suppose r and r' are TP-tables over R, τ. Then the *multiset union of* r *and* r', denoted $r \cup r'$, produces TP-table r'' over R, τ where

$$r'' = \{(d, \Gamma) \mid (d, \Gamma) \in r \,\vee\, (d, \Gamma) \in r'\}$$ $\diamond$

The multiset operators produce TP-tables. TP-tables can be converted into TP-relations through the process of *compaction*. Compactions are performed by executing several instantiations of a *compaction function*. These functions operate by consulting a *combination function*.

Definition 15 (Combination function) Suppose $S = \{[L_1, U_1], \ldots, [L_n, U_n]\}$ is a nonempty multiset of probability intervals and let $[L, U] = ([L_1, U_1] \cap \ldots \cap [L_n, U_n])$. Then χ is a *combination function* if $\chi(S)$ returns a probability interval $[L'', U'']$ that satisfies the following conditions:

1. Identity: If $[L_1, U_1] = \ldots = [L_n, U_n]$, then $[L'', U''] = [L, U]$.
2. Bottomline: $L'' \leq \max_{[L_i, U_i] \in S}(L_i)$.

Furthermore, χ is an *equity combination function* if $[L'', U''] = [L, U]$ whenever $[L, U] \neq \emptyset$. $\diamond$

The following are some sample equity combination functions:

Combination function name	Probability interval returned when $\bigcap_{[L_i, U_i] \in S}[L_i, U_i] = \emptyset$
Optimistic equity	$\chi_{eq}(S) = [\max_{[L_i, U_i] \in S}(L_i), \max_{[L_i, U_i] \in S}(U_i)]$
Enclosing equity	$\chi_{ec}(S) = [\min_{[L_i, U_i] \in S}(L_i), \max_{[L_i, U_i] \in S}(U_i)]$
Pessimistic equity	$\chi_{ep}(S) = [\min_{[L_i, U_i] \in S}(L_i), \min_{[L_i, U_i] \in S}(U_i)]$
Rejecting equity	$\chi_{er}(S) = [0, 0]$
Skeptical equity	$\chi_{esk}(S) = [0, 1]$
Quasi-independence equity	$\chi_{eqi}(S) = [\Pi_{[L_i, U_i] \in S}(L_i), \Pi_{[L_i, U_i] \in S}(U_i)]$

Definition 16 (Compaction function) Suppose G is a nonempty multiset of TP-case statements, t is a time point, and χ is a combination function. Then the multiset $S_{G,t}$ is defined as

$$S_{G,t} = \{\Pr(\gamma, t) \mid (\exists\, \Gamma \in G)\,(\gamma \in \Gamma \,\wedge\, t \in \mathrm{sol}(\gamma.C))\}$$

A function $cmp(G, \chi)$ is a *compaction function* if it returns a TP-case statement Γ'' where

1. $\mathrm{sol}(\bigvee_{\gamma'' \in \Gamma''}(\gamma''.C)) \subseteq \mathrm{sol}(\bigvee_{\gamma' \in \{\gamma \mid (\exists \Gamma \in G)\,(\gamma \in \Gamma)\}}(\gamma'.C))$ and
2. $(\forall \gamma'' \in \Gamma'')\,(\forall t \in \mathrm{sol}(\gamma''.C))\,(\Pr(\gamma'', t) = \chi(S_{G,t}))$. $\diamond$

A simple example is $cmp_s(G, \chi) = \{\langle (t), (t), L, U, u\rangle \mid S_{G,t} \neq \emptyset \,\wedge\, [L, U] = \chi(S_{G,t})\}$.

If δ is a pair of PDFs, then another example is $cmp_p(G, \chi) = \{\langle C'', D'', L'', U'', (\wp''_L, \wp''_U)\rangle\}$ where $C'' = (\bigwedge_{\gamma' \in \{\gamma \mid (\exists \Gamma \in G)\,(\gamma \in \Gamma)\}}(\gamma'.C))$, $D'' = C''$, $L'' = \max_{t \in \mathrm{sol}(D'')}(\max_{[L_i, U_i] \in S_{G,t}}(L_i))$, $U'' = \max_{t \in \mathrm{sol}(D'')}(\max_{[L_i, U_i] \in S_{G,t}}(U_i))$, and $\wp''_L, \wp''_U$ are new PDFs such that for each $t \in \mathrm{sol}(D'')$, $[\wp''_L(D'', t), \wp''_U(D'', t)] = [\ell_t, u_t]$, $\ell_t = \frac{L_t}{L''}$ if $L'' \neq 0$ or $\ell_t = 0$ otherwise, $u_t = \frac{U_t}{U''}$ if $U'' \neq 0$ or $u_t = 0$ otherwise, and $[L_t, U_t] = \chi(S_{G,t})$.

If δ is a single PDF, then a hybrid example is cmp_h. Specifically, compaction function $cmp_h(G, \chi)$ is defined as $\{\langle C'', D'', L'', L'' \cdot x, \wp''_L\rangle\}$ if $cmp_p(G, \chi) = \{\langle C'', D'', L'', U'', (\wp''_L, \wp''_U)\rangle\}$ and $(\exists\, x \in [1, \frac{1}{L''}])\,(\forall t \in \mathrm{sol}(D''))\,(\wp''_L(D'', t) = \wp''_U(D'', t) \cdot x)$. Otherwise, $cmp_h(G, \chi) = cmp_s(G, \chi)$.

Definition 17 (Compaction of a TP-table) Suppose r is a TP-table over R, τ, χ is a combination function, and cmp is a compaction function. Then the *compaction of r under χ using cmp*, denoted $\kappa^{cmp}_{\chi}(r)$, produces TP-relation r'' over R, τ where

$$r'' = \{(d, \Gamma'') \mid (\exists\, \Gamma)\,((d, \Gamma) \in r \,\wedge\, \Gamma'' \neq \emptyset \,\wedge\, \Gamma'' = cmp(\{\Gamma' \mid (d, \Gamma') \in r\}, \chi))\} \qquad \diamond$$

Since $\kappa^{cmp}_{\chi}(r) \equiv \kappa^{cmp'}_{\chi}(r)$ must hold, we let $\kappa_{\chi}(r)$ denote $\kappa^{cmp}_{\chi}(r)$ with any choice for cmp.

Definition 18 (difference between two TP-relations) Suppose r and r' are TP-relations over R, τ. Then the *difference between r and r'*, denoted $r - r'$, produces TP-relation r'' over R, τ where

$$\begin{aligned} r'' = \{(d, \Gamma'') \mid (\exists\, \Gamma)\,(\exists\, \Gamma')\,((d, \Gamma) \in r \,\wedge\, (d, \Gamma') \in r' \,\wedge\, \Gamma'' \neq \emptyset \,\wedge \\ \Gamma'' = \{\langle (C \wedge \neg C'), D, L, U, \delta\rangle \mid \langle C, D, L, U, \delta\rangle \in \Gamma \,\wedge \\ \langle C', D', L', U', \delta'\rangle \in \Gamma' \,\wedge\, \mathrm{sol}(C \wedge \neg C') \neq \emptyset\})\} \end{aligned} \qquad \diamond$$

We are now ready to define a TP-algebra query.

Definition 19 (TPA-query) Suppose db is a TP-database over τ and let q_i denote a TPA-query over db where the answer to q_i is a TP-relation r_i over R_i, τ. Then a *TPA-query over db* has one of the following forms:

1. r where r is a TP-relation in db.
2. $\sigma_{\mathcal{C}}(q_1)$ where $\mathcal{C}$ is a selection condition over R_1, τ.
3. $\pi_{\mathcal{F}}(q_1)$ where P_1 is the primary key for R_1 and $\mathcal{F}$ is an attribute list over R_1, P_1.

4. $\rho_{\mathcal{R}}(q_1)$ where $\mathcal{R}$ is a renaming function over R_1.
5. $\Xi(q_1)$ where Ξ is a TP-compression function.
6. $(q_1 \times_\alpha q_2)$ where α is a probabilistic conjunction strategy and $R_1 \cap R_2 = \emptyset$.
7. $(q_1 \bowtie_\alpha q_2)$ where α is a probabilistic conjunction strategy.
8. $\kappa_\chi(q_1 \cap \ldots \cap q_n)$ where χ is a combination function and $\forall i, j \in [1, n]\, (R_i = R_j)$.
9. $\kappa_\chi(q_1 \cup \ldots \cup q_n)$ where χ is a combination function and $\forall i, j \in [1, n]\, (R_i = R_j)$.
10. $q_1 - q_2$ where $R_1 = R_2$. ◇

Notice that a series of multiset intersections or multiset unions must end with a compaction. This ensures that the answer to a TPA-query is always a TP-relation. Since these operators are often tied together, it is convenient to let $r \cap_\chi r'$ and $r \cup_\chi r'$ denote $\kappa_\chi(r \cap r')$ and $\kappa_\chi(r \cup r')$ respectively.

3 TP-Calculus

We now introduce the temporal probabilistic calculus (TP-calculus). This calculus is similar in spirit to the safe tuple relational calculus [4].

3.1 Syntax of the TP-Calculus

Definition 20 (TP-variable) Suppose $R = (A_1, \ldots, A_k)$ is a relation scheme and $\mathcal{S}_\tau$ is the set of all temporal constraints over τ. Then a *TP-variable over* R, τ is a variable s over the domain

$$dom(s) = dom(A_1) \times \cdots \times dom(A_k) \times dom(C) \times dom(D) \times dom(L) \times dom(U) \times dom(\delta)$$

where $dom(C) = dom(D) = \mathcal{S}_\tau$, $dom(L) = dom(U) = [0, 1]$, and $dom(\delta)$ is the set of all restricted PDFs over calendar τ. Each $\langle d, C, D, L, U, \delta \rangle \in dom(s)$ is an *instance* of s. ◇

We shall abuse notation and write $R \subseteq R'$ to indicate that $A \in R'$ must hold for all $A \in R$.

Definition 21 (TP-atom) Suppose s is a TP-variable over R, τ, s' is a TP-variable over R', τ, $R \subseteq R'$, and $\Theta \in \{<, \leq, =, \neq, \geq, >\}$. Then a *TP-atom over* s, s', τ has one of the following forms:

1. $s.A\ \Theta\ c$ where $A \in R$ and constant $c \in dom(A)$.
2. $s.C : s'.C\ @\ T$ where T is a temporal constraint over τ.
3. $s.C : s'.P\ \Theta\ p$ where $P \in \{L, U\}$ and probability $p \in [0, 1]$.
4. $s.A = s'.A'$ where $A \in R$, $A' \in R'$, and $dom(A) = dom(A')$. ◇

For example, $s.A_1 > 20$ and $s.C : s_1.C\ @\ (1997 \sim 1999)$ are TP-atoms over s, s_1, τ. TP-atoms are used to construct more complex *TP-expressions*.

Definition 22 (Limited TP-expression) Suppose s is a TP-variable over R, τ and s' is a TP-variable over R', τ. Then a *TP-expression over* s, s', τ is defined in the following way:

1. A TP-atom over s, s', τ is a TP-expression over s, s', τ.
2. If E_1, E_2 are TP-expressions over s, s', τ, then $E_1 \wedge E_2$ is a TP-expression over s, s', τ.

A TP-expression E is *limited* if (i) it contains at least one TP-atom of the form $s.A = s'.A'$, and (ii) for all $A' \in R'$, if $s.A_i = s'.A'$ and $s.A_j = s'.A'$ are TP-atoms in E, then $i = j$. $\diamond$

For example, $s.A_1 \leq 3 \wedge s.C : s_1.C$ @ $(1996 \sim 2001) \wedge s.C : s_1.L > .1 \wedge s.A_1 = s_1.A_1$ is a TP-expression over s, s_1, τ. The notion of a *TP-linker* specifies equality constraints that tie together the data attributes of three TP-variables.

Definition 23 (TP-linker) Suppose s, s', and s'' are TP-variables over R, τ, R', τ, and R'', τ respectively. Then a *TP-linker over* s, s', s'' has one of the following forms:

1. $s.A = s'.A'$ if $A \in R$, $A' \in R'$, and $dom(A) = dom(A')$.
2. $s.A = s''.A''$ if $A \in R$, $A'' \in R''$, and $dom(A) = dom(A'')$.
3. $\mathcal{L}_1 \wedge \mathcal{L}_2$ if $\mathcal{L}_1$ and $\mathcal{L}_2$ are TP-linkers over s, s', s''. $\diamond$

For example, $s.A_1 = s'.A_1 \wedge s.A_1 = s''.A_1$ is a TP-linker over s, s', s''.

Definition 24 (Strategy set) A *strategy pair* is an expression of the form $\langle type, str \rangle$ where $type \in \{\otimes, \Xi, \kappa\}$ and str is a probabilistic conjunction strategy when $type = \otimes$, str is a TP-compression function when $type = \Xi$, and str is a combination function when $type = \kappa$.

A *strategy set* is a finite set Ω of strategy pairs such that $\langle type_1, str_1 \rangle \in \Omega$ and $\langle type_2, str_2 \rangle \in \Omega$ implies that $type_1 \neq type_2$. $\diamond$

For example, $\{\langle \otimes, \otimes_{in} \rangle, \langle \Xi, \Xi_{mc} \rangle\}$ and $\{\langle \kappa, \chi_{eq} \rangle, \langle \Xi, \Xi_{mc} \rangle\}$ are strategy sets. We need strategy sets because *TP-formulae* (defined below) are complex formulae that represent queries about conjunctions and/or disjunctions of events. When expressing such queries, the user needs to specify information about (i) their knowledge of the dependencies (if any) between events handled by the query, (ii) whether they want the answer compressed (if so, then how?), and (iii) what combination strategy to use (e.g., to eliminate conflicts).

Definition 25 (TP-formula) Suppose db is a TP-database over τ, s is a TP-variable over R, τ, and P is a primary key for R. Then a *TP-formula over* db has one of the following forms:

1. $\exists\, s\, (s \,\underline{\in}\, r)$ where r is a TP-relation over R, τ and r is in db.
2. $\exists\, s\, (\exists\, s_1\, (F_1' \wedge E))$ where s_1 is a TP-variable over R_1, τ, F_1 is a TP-formula over db of the form $\exists\, s_1\, (F_1')$, E is a *limited* TP-expression over s, s_1, τ, and $\forall a \in P\, (a \in R_1)$.
3. $\exists\, s\, (\exists\, s_1\, (F_1' \wedge_\Omega \exists\, s_2\, (F_2' \wedge \mathcal{L})))$ where F_1 is a TP-formula over db of the form $\exists\, s_1\, (F_1')$, F_2 is a TP-formula over db of the form $\exists\, s_2\, (F_2')$, $\mathcal{L}$ is a TP-linker over s, s_1, s_2, and Ω is a strategy set of the form $\{\langle \otimes, \alpha \rangle, \langle \Xi, \beta \rangle\}$.
4. $\exists\, s\, (\exists\, s_1\, (F_1')\ \theta\ \ldots\ \theta\ \exists\, s_n\, (F_n'))$ where $F_1, \ldots, F_n$ are TP-formulae over db, each F_i has the form $\exists\, s_i\, (F_i')$, each s_i is a TP-variable over R, τ, Ω is a strategy set of the form $\{\langle \kappa, \chi \rangle, \langle \Xi, \beta \rangle\}$, and $\theta \in \{\wedge_\Omega, \vee_\Omega, \wedge\neg\}$. $\diamond$

Example 4 (TP-formulae) Suppose TP-database db contains TP-relations r_1, r_2, r_3 over R_1, R_2, R_3 and τ where $R_1 = (A_1, A_2)$, $R_2 = (A_1, A_3)$, $R_3 = (A_1, A_4)$, and τ is a Gregorian calendar with a chronon of one year. Then the following examples are TP-formulae over db:

1. $\exists\, s_1\, (s_1 \,\underline{\varepsilon}\, r_1)$
   We emphasize that the symbol $\underline{\varepsilon}$ is different from the set membership symbol $\in$. $s_1 \,\underline{\varepsilon}\, r_1$ restricts $dom(s_1)$ in the following way: For each instance $\langle d, C, D, L, U, \delta\rangle \in dom(s_1)$, there must be a TP-tuple $(d, \Gamma) \in r$ where TP-case $\langle C, D, L, U, \delta\rangle \in \Gamma$.
2. $\exists\, s\, (\exists\, s_1\, (s_1 \,\underline{\varepsilon}\, r_1 \,\wedge\, s.C : s_1.L > .1 \,\wedge\, s.A_1 = s_1.A_1))$
   This TP-formula is related to a probabilistic selection followed by a projection that causes s to be a TP-variable over relation scheme (A_1) and calendar τ. To avoid the need for projection, we could add TP-atom $s.A_2 = s_1.A_2$ to the TP-expression used in this TP-formula.
3. $\exists\, s\, (\exists\, s_1\, (s_1 \,\underline{\varepsilon}\, r_1 \,\wedge_{\{\langle\otimes,\otimes_{ig}\rangle,\langle\Xi,\Xi_{mc}\rangle\}}$
   $\exists\, s_2\, (s_2 \,\underline{\varepsilon}\, r_2 \,\wedge\, s.A_1 = s_1.A_1 \,\wedge\, s.A_2 = s_1.A_2 \,\wedge\, s.A_1 = s_2.A_1 \,\wedge\, s.A_3 = s_2.A_3)))$
   This TP-formula is related to a join of r_1 and r_2 under the probabilistic conjunction strategy $\otimes_{ig}$ followed by a TP-compression that uses the Ξ_{mc} strategy. If we want a Cartesian product, then all we need to do is replace TP-linker $s.A_1 = s_2.A_1$ with TP-linker $s.A_1' = s_2.A_1$ in the TP-formula above (this renames A_1 in R_2 to A_2'). To avoid compression, we could replace Ξ_{mc} with the identity TP-compression function Ξ_{id}.
4. $\exists\, s\, (\exists\, s_1\, (\exists\, s_1'\, (s_1' \,\underline{\varepsilon}\, r_1 \,\wedge\, s_1.A_1 = s_1'.A_1)) \,\wedge_{\{\langle\kappa,\chi_{eq}\rangle,\langle\Xi,\Xi_{mc}\rangle\}}$
   $\exists\, s_2\, (\exists\, s_2'\, (s_2' \,\underline{\varepsilon}\, r_2 \,\wedge\, s_2.A_1 = s_2'.A_1)) \,\wedge_{\{\langle\kappa,\chi_{eq}\rangle,\langle\Xi,\Xi_{mc}\rangle\}} \exists\, s_3\, (\exists\, s_3'\, (s_3' \,\underline{\varepsilon}\, r_3 \,\wedge\, s_3.A_1 = s_3'.A_1)))$
   This TP-formula is related to the multiset intersection of $\pi_{A_1}(r_1)$, $\pi_{A_1}(r_2)$, and $\pi_{A_1}(r_3)$ followed by a compaction that uses the χ_{eq} combination function and a TP-compression that uses the Ξ_{mc} strategy. Notice that unless $dom(A_2) = dom(A_3) = dom(A_4)$, the fourth rule for constructing TP-formulae will not allow us to intersect r_1, r_2, and r_3. ◇

We are now ready to define a TP-calculus query.

Definition 26 (TPC-query) Suppose db is a TP-database over τ and F is a TP-formula of the form $\exists\, s\, (F')$ where (i) s is the only free variable in F' and (ii) $r \in db$ for every TP-relation r mentioned in F. Then $\{s \mid F'\}$ is a *TPC-query over db*. ◇

We will assume throughout this paper that all quantified TP-variables in a TPC-query differ from each other and from the lone free TP-variable. There is no loss of generality in making this assumption since TP-variables can easily be renamed.

3.2 Semantics of the TP-calculus

In this section, we provide a quick semantics for TPC-queries. We start with the definition of a TP-assignment which is similar to the concept of a variable assignment in classical logic [29].

Definition 27 (TP-assignment) A *TP-assignment* is a function $\mathcal{A}$ that maps each TP-variable s to an instance $\langle d_s^{\mathcal{A}}, C_s^{\mathcal{A}}, D_s^{\mathcal{A}}, L_s^{\mathcal{A}}, U_s^{\mathcal{A}}, \delta_s^{\mathcal{A}}\rangle \in dom(s)$. We omit the superscipt $\mathcal{A}$ when it is clear from context. Also, we let γ_s denote the TP-case $\langle C_s, D_s, L_s, U_s, \delta_s\rangle$. ◇

For example, $\mathcal{A}$ may assign s to a data tuple d and a TP-case $\gamma \in \Gamma$ where $(d, \Gamma) \in r$ for some TP-relation r. The following definition specifies when a TP-assignment satisfies a TP-expression.

Definition 28 (Suitable/satisfying TP-assignment) Suppose $\mathcal{A}$ is a TP-assignment, s is a TP-variable over R, τ, s' is a TP-variable over R', τ, and E is a TP-expression over s, s', τ. Then $\mathcal{A}$ *is suitable for* E, denoted $\mathcal{A} \rhd E$, if $\emptyset \subset \mathrm{sol}(C_s) \subseteq \mathrm{sol}(C_{s'})$ and the following constraints are satisfied:

1. If E has the form $s.A \; \Theta \; c$, then $(d_s.A \; \Theta \; c)$.
2. If E has the form $s.C : s'.C \; @ \; T$, then $\forall t \in \mathrm{sol}(C_s)(t \in \mathrm{sol}(C_{s'} \wedge T))$.
3. If E has the form $s.C : s'.P \; \Theta \; p$, then $\forall t \in \mathrm{sol}(C_s)(\delta_{s'}(D_{s'}, t) \cdot P_{s'} \; \Theta \; p)$.
4. If E has the form $s.A = s'.A'$, then $(d_s.A = d_{s'}.A')$.
5. If E has the form $E_1 \wedge E_2$, then $\mathcal{A} \rhd E_1$ and $\mathcal{A} \rhd E_2$.

We say that $\mathcal{A}$ *satisfies* E, denoted $\mathcal{A} \models E$, iff $\mathcal{A} \rhd E$ and the following constraints are satisfied:

1. $R = \{A \mid$ there exists a TP-atom of the form $s.A = s'.A'$ in $E\}$. We assume here that relation scheme R is treated as a set, i.e., the attribute ordering does not matter.
2. There is no temporal constraint C where $\mathrm{sol}(C) \supset \mathrm{sol}(C_s)$ and replacing C_s above with C allows $\mathcal{A}$ to be suitable for E.
3. $\langle D_s, L_s, U_s, \delta_s \rangle = \langle D_{s'}, L_{s'}, U_{s'}, \delta_{s'} \rangle$. ◇

Example 5 (Satisfying TP-assignment) Suppose $s_1 \underline{\in} \mathsf{TrainDep}$ (see Example 1). Then

$$
\begin{aligned}
dom(s_1) = \{&\langle 151, \text{Baltimore}, \text{New York}, (12{:}05 \sim 12{:}14), (12{:}05 \sim 12{:}14), 0.5, 0.6, u \rangle, \\
&\langle 151, \text{Baltimore}, \text{New York}, (12{:}15 \sim 12{:}20), (12{:}15 \sim 12{:}20), 0.3, 0.4, g, 0.5 \rangle\}
\end{aligned}
$$

Furthermore, suppose $s.C : s_1.C \; @ \; (12{:}12 \sim 12{:}22) \wedge s.C : s_1.L > .04 \wedge s.\mathsf{TrainNo} = s_1.\mathsf{TrainNo}$ is TP-expression E. Then $\mathcal{A} \models E$ if TP-assignment $\mathcal{A}$ satisfies one of the following conditions:

- $\mathcal{A}(s) = \langle 151, (12{:}12 \sim 12{:}14), (12{:}05 \sim 12{:}14), 0.5, 0.6, u \rangle$ and
  $\mathcal{A}(s_1) = \langle 151, \text{Baltimore}, \text{New York}, (12{:}05 \sim 12{:}14), (12{:}05 \sim 12{:}14), 0.5, 0.6, u \rangle$.
- $\mathcal{A}(s) = \langle 151, (12{:}15 \sim 12{:}16), (12{:}15 \sim 12{:}20), 0.3, 0.4, g, 0.5 \rangle$ and
  $\mathcal{A}(s_1) = \langle 151, \text{Baltimore}, \text{New York}, (12{:}15 \sim 12{:}20), (12{:}15 \sim 12{:}20), 0.3, 0.4, g, 0.5 \rangle$. ◇

Recall that $\mathcal{S}_\tau$ denotes the set of all temporal constraints over τ. Consider a partitioning of $\mathcal{S}_\tau$ where C and C' are in the same partition iff $\mathrm{sol}(C) = \mathrm{sol}(C')$. Although the size of each partition is infinite, we can restrict ourselves to use only one *canonical* temporal constraint for each partition. It is important to note that under this restriction, the set of all TP-assignments $\mathcal{A}$ where $\mathcal{A} \models E$ is finite if E is a limited TP-expression over s, s_1, τ and $dom(s_1)$ is finite.

Definition 29 (TP-linker satisfaction) Suppose $\mathcal{A}$ is a TP-assignment, s is a TP-variable over R, τ, s' is a TP-variable over R', τ, s'' is a TP-variable over R'', τ, and $\mathcal{L}$ is a TP-linker over s, s', s''. Then $\mathcal{A}$ *satisfies* $\mathcal{L}$, denoted $\mathcal{A} \models \mathcal{L}$, iff the following constraints are satisfied:

1. $R = \{A \mid$ there exists a TP-linker of the form $s.A = s'.A'$ or $s.A = s''.A''$ in $\mathcal{L}\}$. As in the preceding definition, we assume that the attribute ordering is not important.
2. If $\mathcal{L}$ is a TP-expression E_1 over s, s', τ, then $\mathcal{A} \triangleright E_1$.
3. If $\mathcal{L}$ is a TP-expression E_2 over s, s'', τ, then $\mathcal{A} \triangleright E_2$.
4. If $\mathcal{L} = \mathcal{L}_1 \wedge \mathcal{L}_2$ where $\mathcal{L}_1$ is a TP-expression E_1 over s, s', τ and $\mathcal{L}_2$ is a TP-expression E_2 over s, s'', τ, then $\mathcal{A} \triangleright E_1$ and $\mathcal{A} \triangleright E_2$. ◇

For example, suppose $s_1 \underline{\in} \mathsf{TrainDep}$, $s_2 \underline{\in} \mathsf{BusArr}$, and $\mathcal{L}$ is the following TP-linker over s, s_1, s_2:

$$s.\mathsf{TrainNo} = s_1.\mathrm{TrainNo} \wedge s.\mathsf{TrainFrom} = s_1.\mathrm{From} \wedge s.\mathsf{TrainTo} = s_1.\mathrm{To} \wedge$$
$$s.\mathsf{BusNo} = s_2.\mathrm{BusNo} \wedge s.\mathsf{BusFrom} = s_2.\mathrm{From} \wedge s.\mathsf{BusTo} = s_2.\mathrm{To}$$

Furthermore, suppose s is a TP-variable over R, τ. Then $\mathcal{A} \models \mathcal{L}$ if (i) the set for relation scheme R is $\{\mathsf{TrainNo}, \mathsf{TrainFrom}, \mathsf{TrainTo}, \mathsf{BusNo}, \mathsf{BusFrom}, \mathsf{BusTo}\}$ and (ii) $d_s^{\mathcal{A}}.A = d_{s_i}^{\mathcal{A}}.A'$ for each TP-linker in $\mathcal{L}$ of the form $s.A = s_i.A'$ where $i \in \{1, 2\}$.

Definition 30 (TP-formula semantics) Suppose $\mathcal{A}$ is a TP-assignment and F is a TP-formula over db. Then $\mathcal{A}$ *is a model for* F, denoted $\mathcal{A} \models F$, iff the following constraints are satisfied:

1. If F has the form $\exists\, s\,(s \underline{\in} r)$, then there exists a $(d_s, \Gamma) \in r$ where $\langle C_s, D_s, L_s, U_s, \delta_s\rangle \in \Gamma$.
2. If F has the form $\exists\, s\,(\exists\, s_1\,(F_1' \wedge E))$, then $\mathcal{A} \models \exists\, s_1\,(F_1')$ and $\mathcal{A} \models E$.
3. If F has the form $\exists\, s\,(\exists\, s_1\,(F_1' \wedge_\Omega \exists\, s_2\,(F_2' \wedge \mathcal{L})))$ where $\Omega = \{\langle\otimes, \alpha\rangle, \langle\Xi, \beta\rangle\}$, then
   - $\mathcal{A} \models \exists\, s_1\,(F_1')$, $\mathcal{A} \models \exists\, s_2\,(F_2')$, $\mathcal{A} \models \mathcal{L}$, $\mathrm{sol}(C_s) \neq \emptyset$, and $C_s = (C_{s_1} \wedge C_{s_2})$.
   - $\forall t \in \mathrm{sol}(C_s)\,(\Pr(\gamma_s, t) = \Pr(\gamma_{s_1}, t) \otimes_\alpha \Pr(\gamma_{s_2}, t))$.
4. If F has the form $\exists\, s\,(\exists\, s_1\,(F_1')\ \theta\ \ldots\ \theta\ \exists\, s_n\,(F_n'))$ where $\Omega = \{\langle\kappa, \chi\rangle, \langle\Xi, \beta\rangle\}$, then
   - $\forall i \in [1, n]\,(\mathcal{A} \models \exists\, s_i\,(F_i'))$ and $\mathrm{sol}(C_s) \neq \emptyset$.
   - If $\theta = \wedge_\Omega$, then $d_s = d_{s_1} = \ldots = d_{s_n}$, $C_s = (C_{s_1} \wedge \ldots \wedge C_{s_n})$, and $\forall t \in \mathrm{sol}(C_s)\,(\Pr(\gamma_s, t) = \chi(\{\Pr(\gamma_{s_1}, t), \ldots, \Pr(\gamma_{s_n}, t)\}))$.
   - If $\theta = \vee_\Omega$, then $d_s \in \{d_{s_1}, \ldots, d_{s_n}\}$, $C_s = (t)$, and $\Pr(\gamma_s, t) = \chi(\{\Pr(\gamma_{s_i}, t) \mid i \in I_t\})$ where $I_t = \{i \mid \exists\, \mathcal{A}'\,(\mathcal{A}' \models \exists\, s_i(F_i') \wedge (d_{s_i}^{\mathcal{A}'} = d_s) \wedge t \in \mathrm{sol}(C_{s_i}^{\mathcal{A}'}))\}$.
   - If $\theta = \wedge\neg$, then $d_s = d_{s_1}$ and $\gamma_s = \langle(C_{s_1} \wedge \neg\, C'), D_{s_1}, L_{s_1}, U_{s_1}, \delta_{s_1}\rangle$ where $C' = \bigvee_{i=2}^{n} \{C \mid \exists\, \mathcal{A}'\,(\mathcal{A}' \models \exists\, s_i(F_i') \wedge (d_{s_i}^{\mathcal{A}'} = d_s) \wedge (C_{s_i}^{\mathcal{A}'} = C))\}$. ◇

Definition 31 (TPC-query semantics) Suppose F is a TP-formula of the form $\exists\, s\,(F')$. Then the *answer* to TPC-query $\{s \mid F'\}$ is defined as the following TP-table:

$$\{(d, \Gamma) \mid \exists\, \mathcal{A}\,(\mathcal{A} \models F \wedge d = d_s^{\mathcal{A}} \wedge \Gamma = \{\gamma_s^{\mathcal{A}}\})\} \qquad ◇$$

The following important theorems jointly state that the TP-algebra and the TP-calculus have exactly the same expressive power.

Theorem 1 (TPA ⇒ TPC) Every TPA-query can be expressed as a TPC-query. ◇

Theorem 2 (TPC ⇒ TPA) Every TPC-query can be expressed as a TPA-query. ◇

4 Equivalence Results

In this section we describe a set of query equivalence results that hold in the TP-algebra. These query equivalences provide *rewrite rules* that may be used to optimize queries. In the sequel, we assume r, r' are TP-tables, and $\mathcal{C}$ is a selection condition. Recall, from Section 2, that two TP-relations are equivalent if their annotated expansions are identical — informally speaking, two TP-relations are equivalent iff whenever tp is the restriction of a TP-tuple to its data attributes, if the probability that tp is true at time t is p_1 according to the first TP-relation, then the probability that tp is true at time t is p_1 according to the second relation as well, i.e. the two TP-relations assign the same probabilities. Also, unless stated otherwise, the following equivalences hold for *all* combination functions and for *all* probabilistic conjunction strategies, thus making the results very widely applicable.

4.1 Set-Theoretical Properties

The standard relational algebra operations are *idempotent*, *commutative*, and *associative*. These properties do not hold for all TP-algebra operations. Our first result says that selection, projection, compaction, union, and intersection are idempotent, and an idempotence style result holds for difference as well.

Theorem 3 (Idempotence) The following equivalences hold:

1. $\sigma_{\mathcal{C}}(\sigma_{\mathcal{C}}(r)) \equiv \sigma_{\mathcal{C}}(r)$;
2. $\pi_{\mathcal{F}}(\pi_{\mathcal{F}}(r)) \equiv \pi_{\mathcal{F}}(r)$;
3. $\kappa_{\chi}(\kappa_{\chi}(r)) = \kappa(r)$ if χ is any combination function;
4. $r \cap_{\chi} r \equiv r$ if r is a TP-relation and χ is any combination function;
5. $r \cup_{\chi} r \equiv r$ if r is a TP-relation and χ is any combination function;
6. $(r - r') - r' \equiv r - r'$.

The following two results show that most of the important operations in the TP-algebra are commutative, but not all are associative.

Theorem 4 (Commutativity) The following equivalences hold:

1. $\sigma_{\mathcal{C}}(\sigma_{\mathcal{C}'}(r)) \equiv \sigma_{\mathcal{C}'}(\sigma_{\mathcal{C}}(r))$;
2. $\pi_{\mathcal{F}}(\pi_{\mathcal{G}}(r)) \equiv \pi_{\mathcal{F}\cap\mathcal{G}}(r) \equiv \pi_{\mathcal{G}}(\pi_{\mathcal{F}}(r))$;
3. $r \cap_{\chi} r' \equiv r' \cap_{\chi} r$;
4. $r \cup_{\chi} r' \equiv r' \cup_{\chi} r'$;
5. $(r - r') - r'' \equiv (r - r'') - r' \equiv r - (r' \cup r'')$;
6. $r \times_{\alpha} r' \equiv r' \times_{\alpha} r$ (ignoring the order of data attributes);
7. $r \bowtie_{\alpha} r' \equiv r' \bowtie_{\alpha} r$ (ignoring the order of data attributes).

Theorem 5 (Associativity) The following equivalences hold:

1. $(r \times_{\alpha} r') \times_{\alpha} r'' \equiv r \times_{\alpha} (r' \times_{\alpha} r'')$;

2. $(r \bowtie_\alpha r') \bowtie_\alpha r'' \equiv r \bowtie_\alpha (r' \bowtie_\alpha r'')$.

In general, intersection and union are not associative because both these operations involve applying the compaction operator. The following example shows why intersection is not associative.

Example 6 (Intersection is not associative) Recall that for each data tuple d and time point t, the $\kappa_{ec}^{cmp_s}$ operator collects all intervals $[l_1, u_1], \ldots, [l_k, u_k]$ associated with d and t by different TP-tuples and computes the new interval $[l, u]$ as follows:

$$[l, u] = \begin{cases} [l_1, u_1] \cap \ldots \cap [l_k, u_k] & \text{iff } [l_1, u_1] \cap \ldots \cap [l_k, u_k] \neq \emptyset; \\ [\min(l_1, \ldots, l_k), \max(u_1, \ldots, u_k)] & \text{otherwise.} \end{cases}$$

Now consider three TP-relations r_1, r_2 and r_3 such that for some data tuple d and time point t, $(d, \Gamma_1) \in r_1$, $\langle t \sim t, t \sim t, 0.2, 0.4, u\rangle \in \Gamma_1$, $(d, \Gamma_2) \in r_2$, $\langle t \sim t, t \sim t, 0.3, 0.6, u\rangle \in \Gamma_2$, $(d, \Gamma_3) \in r_3$, and $\langle t \sim t, t \sim t, 0.5, 0.7, u\rangle \in \Gamma_3$. Then, $r_1 \cap_{ec} r_2$ will contain (d, Γ_{12}) where TP-case $\langle t \sim t, t \sim t, 0.3, 0.4, u\rangle \in \Gamma_{12}$ since $[0.3, 0.4] = [0.2.0.4] \cap [0.3, 0.6]$. Also, $r_2 \cap_{ec} r_3$ will contain TP-tuple (d, Γ_{23}) where TP-case $\langle t \sim t, t \sim t, 0.5, 0.6, u\rangle \in \Gamma_{23}$ since $[0.5, 0.6] = [0.3.0.6] \cap [0.5, 0.7]$.

But then $(r_1 \cap_{ec} r_2) \cap_{ec} r_3$ will contain TP-tuple $(d, \Gamma_{12,3})$ where $\langle t \sim t, t \sim t, 0.3, 0.7, u\rangle \in \Gamma_{12,3}$ as $[0.3, 0.4] \cap [0.5, 0.7] = \emptyset$ and so the probability interval is $[\min(0.3, 0.5), \max(0.4, 0.7)] = [0.3, 0.7]$. On the other hand, $r_1 \cap_{ec} (r_2 \cap_{ec} r_3)$ will contain TP-tuple $(d, \Gamma_{1,23})$ where $\langle t \sim t, t \sim t, 0.2, 0.6, u\rangle \in \Gamma_{1,23}$ as $[0.2, 0.4] \cap [0.5, 0.6] = \emptyset$ and so the probability interval is $[\min(0.2, 0.5), \max(0.4, 0.6)] = [0.2, 0.6]$. This indicates that $(r_1 \cap_{ec} r_2) \cap_{ec} r_3 \not\equiv r_1 \cap_{ec} (r_2 \cap_{ec} r_3)$.

Proposition 1 The following properties hold:

1. $r \cap_\chi r' \subseteq r \cup_\chi r'$;
2. $r \cap_\chi r' \not\equiv r - (r - r')$;
3. $r - (r' \cap_\chi r'') \equiv (r - r') \cup_\chi (r - r'')$;
4. $r - (r' \cup_\chi r'') \equiv (r - r') \cap_\chi (r - r'')$.

4.2 Pushing Selection Through Other Operations

An important rewrite rule in the classical relational algebra allows us to push selection through some expensive operations like join. In this section, we study the cases where selection can be pushed through various TP-operations. While many results are similar to the corresponding equivalences from the classical relational algebra, in many important cases (e.g., pushing selection into Cartesian product/join), general results can only be obtained for *data* and *temporal* selection conditions. Specific results for probabilistic selection conditions are discussed in Section 4.3.

Theorem 6 (Selection-projection) Suppose $\mathcal{F}$ is an attribute list and $\mathcal{C}$ does not involve any of these attributes. Then $\sigma_C(\pi_{\mathcal{F}}(r)) \equiv \pi_{\mathcal{F}}(\sigma_C(r))$.

The results below hold only for *data* and *temporal* selection conditions. The next result shows that selection can be pushed into a compaction. From an efficiency standpoint, this is good because selection can often be performed much faster than compaction.

Theorem 7 (Pushing selection into compaction) Let $\mathcal{C}$ be a data condition or a temporal condition. Then $\sigma_{\mathcal{C}}(\kappa_{\chi}(r)) \equiv \kappa_{\chi}(\sigma_{\mathcal{C}}(r'))$.

The following theorem shows that in some cases, selections can be pushed into Cartesian products.

Theorem 8 (Pushing selection into Cartesian product) Suppose α is a PCS, $\mathcal{C}_1$ (resp. $\mathcal{C}_2$) is a *data* selection condition for r (resp. r'), and $\mathcal{C}$ is a *temporal* selection condition. Then

1. $\sigma_{\mathcal{C}_1}(r \times_{\alpha} r') \equiv \sigma_{\mathcal{C}_1}(r) \times_{\alpha} r'$.
2. $\sigma_{\mathcal{C}_2}(r \times_{\alpha} r') \equiv r \times_{\alpha} \sigma_{\mathcal{C}_2}(r')$.
3. $\sigma_{\mathcal{C}}(r \times_{\alpha} r') \equiv \sigma_{\mathcal{C}}(r) \times_{\alpha} \sigma_{\mathcal{C}}(r') \equiv r \times_{\alpha} \sigma_{\mathcal{C}}(r') \equiv \sigma_{\mathcal{C}}(r) \times_{\alpha} r'$.

Not surprisingly, a similar theorem holds for join.

Theorem 9 (Pushing selection into join) We use the same notation as Theorem 8.

1. $\sigma_{\mathcal{C}_1}(r \bowtie_{\alpha} r') \equiv \sigma_{\mathcal{C}_1}(r) \bowtie_{\alpha} r'$.
2. $\sigma_{\mathcal{C}_2}(r \bowtie_{\alpha} r') \equiv r \bowtie_{\alpha} \sigma_{\mathcal{C}_2}(r')$.
3. $\sigma_{\mathcal{C}}(r \bowtie_{\alpha} r') \equiv \sigma_{\mathcal{C}}(r) \bowtie_{\alpha} \sigma_{\mathcal{C}}(r') \equiv r \bowtie_{\alpha} \sigma_{\mathcal{C}}(r') \equiv \sigma_{\mathcal{C}}(r) \bowtie_{\alpha} r'$.

Next, we establish equivalences for pushing selections into set operations.

Theorem 10 (Pushing selection into set operations) Let $\mathcal{C}$ be either a data condition or a temporal condition. Then

1. $\sigma_{\mathcal{C}}(r \cap_{\chi} r') \equiv \sigma_{\mathcal{C}}(r) \cap_{\chi} \sigma_{\mathcal{C}}(r') \equiv r \cap_{\chi} \sigma_{\mathcal{C}}(r') \equiv \sigma_{\mathcal{C}}(r) \cap_{\chi} r'$.
2. $\sigma_{\mathcal{C}}(r \cup_{\chi} r') \equiv \sigma_{\mathcal{C}}(r) \cup_{\chi} \sigma_{\mathcal{C}}(r')$.
3. $\sigma_{\mathcal{C}}(r - r') \equiv \sigma_{\mathcal{C}}(r) - \sigma_{\mathcal{C}}(r') \equiv \sigma_{\mathcal{C}}(r) - r'$.

4.3 Conditional Query Equivalences in TPA

In this section, we first (Section 4.3.1) explain why it is hard to find equivalence results probabilistic selections. Then in Section 4.3.2, we provide a number of conditional equivalence results that we have obtained for probabilistic selections under a variety of special cases.

4.3.1 Why equivalences involving probabilistic selection conditions are hard

We start with an example that provides strong evidence indicating that no simple, general rule for pushing probabilistic selections into Cartesian products (and hence, joins) exists.

Example 7 (Probabilistic selection and Cartesian product) Consider TP-relations r_1 and r_2 of Example 2 and suppose that their attributes have been renamed via the renaming functions from Example 3. Suppose probabilistic selection condition $\mathcal{C}^* = (L \leq 0.04) \vee (U \in [0.1, 0.15])$. Then Table 1 shows the results of the following four queries:

$d =$

BusNo	BusFrom	BusTo	TrainNo	TrainFrom	TrainTo
23	Rockville	Baltimore	151	Baltimore	New York

$q_1 = \sigma_{(L \leq 0.04) \vee (U \in [0.1, 0.15])}(r_2 \times_{in} r_1)$

Data Part	C	D	L	U	δ
d	(12:15 $\sim$ 12:15)	(12:15 $\sim$ 12:15)	0.01875	0.025	u
	(12:17 $\sim$ 12:17)	(12:17 $\sim$ 12:17)	0.001875	0.0025	u

$q_2 = \sigma_{(L \leq 0.04) \vee (U \in [0.1, 0.15])}(r_2) \times_{in} \sigma_{(L \leq 0.04) \vee (U \in [0.1, 0.15])}(r_1)$

Data Part	C	D	L	U	δ

$q_3 = \sigma_{(L \leq 0.04) \vee (U \in [0.1, 0.15])}(r_2) \times_{in} r_1$

Data Part	C	D	L	U	δ
d	(12:15 $\sim$ 12:15)	(12:15 $\sim$ 12:15)	0.01875	0.025	u

$q_4 = r_2 \times_{in} \sigma_{(L \leq 0.04) \vee (U \in [0.1, 0.15])}(r_1)$

Data Part	C	D	L	U	δ
d	(12 : 17 $\sim$ 12 : 17)	(12 : 17 $\sim$ 12 : 17)	0.001875	0.0025	u

Table 1: Different combinations of Cartesian product and probabilistic selection

$$q_1 = \sigma_{\mathcal{C}^*}(r_2 \times_{in} r_1); \quad q_2 = \sigma_{\mathcal{C}^*}(r_2) \times_{in} \sigma_{\mathcal{C}^*}(r_1);$$
$$q_3 = \sigma_{\mathcal{C}^*}(r_2) \times_{in} r_1; \quad q_4 = r_2 \times_{in} \sigma_{\mathcal{C}^*}(r_1).$$

It is clear from Table 1 that no appropriate rewrite rule similar to those described in Theorem 8 for $\sigma_{\mathcal{C}^*}(r_2 \times_{in} r_1)$ can be obtained as the answers to all four queries are different. We note here that $\sigma_{\mathcal{C}^*}(r_2)$ selects only the time point 12:15 and $\sigma_{\mathcal{C}^*}(r_2)$ selects only the time point 12:17. This makes the result of q_2 an empty set.

Let us now consider an example involving compaction and probabilistic selects.

Example 8 (Probabilistic selection and compaction) Consider the κ_{ec} operator from Example 6 and suppose TP-case $\gamma_1 = \langle t \sim t, t \sim t, 0.3, 0.6, u\rangle$, $\gamma_2 = \langle t \sim t, t \sim t, 0.2, 0.4, u\rangle$, and $\gamma_{12} = \langle t \sim t, t \sim t, 0.3, 0.4, u\rangle$. Also, suppose TP-table $r = \{(d, \Gamma_1), (d, \Gamma_2)\}$ where $\Gamma_1 = \{\gamma_1\}$ and $\Gamma_2 = \{\gamma_2\}$. Then $\kappa_{ec}(r) = \{(d, \Gamma_{12})\}$ where $\Gamma_{12} = \{\gamma_{12}\}$ since $[0.3, 0.4] = [0.3, 0.6] \cap [0.2, 0.4]$.

Consider the probabilistic selection condition $U = 0.4$. We see that TP-case γ_{12} is in the answer to $\sigma_{U=0.4}(\kappa_{ec}(r))$ since its upper bound is 0.4. In contrast, the answer to $\sigma_{U=0.4}(r)$ contains γ_2 but not γ_1 so the answer to $\kappa_{ec}(\sigma_{U=0.4}(r))$ can only contain γ_2. Therefore, $\sigma_{U=0.4}(\kappa_{ec}(r))$ and $\kappa_{ec}(\sigma_{U=0.4}(r))$ are not equivalent.

As compaction is used to define both intersection and union, probabilistic selections and intersections/unions do not commute. The next example shows that $\sigma_{\mathcal{C}}(r - r')$ is not equivalent to $\sigma_{\mathcal{C}}(r) - \sigma_{\mathcal{C}}(r')$ when $\mathcal{C}$ is a probabilistic selection condition.

Example 9 (Probabilistic selection and difference) Suppose TP-relation $r_1 = (d, \{\gamma_1\})$ and TP-relation $r_2 = (d, \{\gamma_2\})$ where $\gamma_1 = \langle t \sim t, t \sim t, 0.4, 0.6, u\rangle$ and $\gamma_2 = \langle t \sim t, t \sim t, 0.5, 0.6, u\rangle$. Then $r_1 - r_2 = \emptyset$ so $\sigma_{L=0.4}(r_1 - r_2) = \emptyset$. On the other hand, $\sigma_{L=0.4}(r_1)$ will contain γ_1 and $\sigma_{L=0.4}(r_2)$ will not contain γ_2 so $\sigma_{L=0.4}(r_1) - \sigma_{L=0.4}(r_2)$ will contain γ_1. Thus, $\sigma_{L=0.4}(r_1 - r_2) \not\equiv \sigma_{L=0.4}(r_1) - \sigma_{L=0.4}(r_2)$.

4.3.2 Specific query equivalences for probabilistic selections

Though the negative results about probabilistic selection presented above are discouraging, there is good news: In some cases, probabilistic selection can be pushed into other operations. In our first result, we consider only probabilistic selection conditions of the form $P > p$ or $P \geq p$.

Theorem 11 (Pushing selections of the form $P > p$ or $P \geq p$) Suppose $\mathcal{C}$ is a probabilistic selection condition of the form $L > p$, $U > p$, $L \geq p$, or $U \geq p$. Then

1. $\sigma_{\mathcal{C}}(r \times_{\alpha} r') \equiv \sigma_{\mathcal{C}}(\sigma_{\mathcal{C}}(r) \times_{\alpha} \sigma_{\mathcal{C}}(r'))$;
2. $\sigma_{\mathcal{C}}(r \bowtie_{\alpha} r') \equiv \sigma_{\mathcal{C}}(\sigma_{\mathcal{C}}(r) \bowtie_{\alpha} \sigma_{\mathcal{C}}(r'))$.

The following theorem indicates that when the probabilistic selection condition has the form $P < p$ or $P \leq p$ and we use the *positive correlation* PCS, then we can push probabilistic selection into Cartesian product and join.

Theorem 12 (Pushing selection into Cartesian product and join under the $\otimes_{pc}$ PCS) : Suppose $\mathcal{C}$ is a probabilistic selection condition of the form $L < p$, $U < p$, $L \leq p$, or $U \leq p$ and suppose χ is any equity combination function. Then

1. $\sigma_{\mathcal{C}}(r \times_{pc} r') \equiv (\sigma_{\mathcal{C}}(r) \times_{pc} r') \cup_{\chi} (r \times_{pc} \sigma_{\mathcal{C}}(r'))$;
2. $\sigma_{\mathcal{C}}(r \bowtie_{pc} r') \equiv (\sigma_{\mathcal{C}}(r) \bowtie_{pc} r') \cup_{\chi} (r \bowtie_{pc} \sigma_{\mathcal{C}}(r'))$.

When the selection condition has the form $P < p$ or $P \leq p$ and we use the *independence* PCS, then the following theorem indicates that we can push probabilistic selection into Cartesian product and join if the optimizer stores the statistics MIN_L(r) and MIN_U(r) for each TP-relation r. As we shall see in Section 5, MIN_L(r) is defined as $\min\{l \mid l = \delta(D,t) \cdot L \wedge \langle C, D, L, U, \delta\rangle \in \Gamma \wedge (d, \Gamma) \in r\}$ and MIN_U(r) is defined as $\min\{u \mid u = \delta(D,t) \cdot U \ \wedge \ \langle C, D, L, U, \delta\rangle \in \Gamma \ \wedge \ (d, \Gamma) \in r\}$.

Theorem 13 (Pushing selection into Cartesian product under the $\otimes_{in}$ PCS) :

1. $\sigma_{L \leq p}(r \times_{in} r') \equiv \sigma_{L \leq p}(r \times_{in} \sigma_{L \leq \frac{p}{\text{MIN\_L}(r)}}(r')) \equiv$
   $\sigma_{L \leq p}(\sigma_{L \leq \frac{p}{\text{MIN\_L}(r')}}(r) \times_{in} \sigma_{L \leq \frac{p}{\text{MIN\_L}(r)}}(r')) \equiv \sigma_{L \leq p}(\sigma_{L \leq \frac{p}{\text{MIN\_L}(r')}}(r) \times_{in} r')$;
2. $\sigma_{L < p}(r \times_{in} r') \equiv \sigma_{L < p}(r \times_{in} \sigma_{L < \frac{p}{\text{MIN\_L}(r)}}(r')) \equiv$
   $\sigma_{L < p}(\sigma_{L < \frac{p}{\text{MIN\_L}(r')}}(r) \times_{in} \sigma_{L < \frac{p}{\text{MIN\_L}(r)}}(r')) \equiv \sigma_{L < p}(\sigma_{L < \frac{p}{\text{MIN\_L}(r')}}(r) \times_{in} r')$;
3. $\sigma_{U \leq p}(r \times_{in} r') \equiv \sigma_{U \leq p}(r \times_{in} \sigma_{U \leq \frac{p}{\text{MIN\_U}(r)}}(r')) \equiv$
   $\sigma_{U \leq p}(\sigma_{U \leq \frac{p}{\text{MIN\_U}(r')}}(r) \times_{in} \sigma_{U \leq \frac{p}{\text{MIN\_U}(r)}}(r')) \equiv \sigma_{U \leq p}(\sigma_{U \leq \frac{p}{\text{MIN\_U}(r')}}(r) \times_{in} r')$;
4. $\sigma_{U < p}(r \times_{in} r') \equiv \sigma_{U < p}(r \times_{in} \sigma_{U < \frac{p}{\text{MIN\_U}(r)}}(r')) \equiv$
   $\sigma_{U < p}(\sigma_{U < \frac{p}{\text{MIN\_U}(r')}}(r) \times_{in} \sigma_{U < \frac{p}{\text{MIN\_U}(r)}}(r')) \equiv \sigma_{U < p}(\sigma_{U < \frac{p}{\text{MIN\_U}(r')}}(r) \times_{in} r')$.

An analogous equivalence holds for join as shown below.

Corollary 1 (Pushing selection into join under the $\otimes_{in}$ PCS) :

1. $\sigma_{L\leq p}(r \bowtie_{in} r') \equiv \sigma_{L\leq p}(r \bowtie_{in} \sigma_{L\leq \frac{p}{\text{MIN\_L}(r)}}(r')) \equiv$
   $\sigma_{L\leq p}(\sigma_{L\leq \frac{p}{\text{MIN\_L}(r')}}(r) \bowtie_{in} \sigma_{L\leq \frac{p}{\text{MIN\_L}(r)}}(r')) \equiv \sigma_{L\leq p}(\sigma_{L\leq \frac{p}{\text{MIN\_L}(r')}}(r) \bowtie_{in} r')$;
2. $\sigma_{L<p}(r \bowtie_{in} r') \equiv \sigma_{L<p}(r \bowtie_{in} \sigma_{L< \frac{p}{\text{MIN\_L}(r)}}(r')) \equiv$
   $\sigma_{L<p}(\sigma_{L< \frac{p}{\text{MIN\_L}(r')}}(r) \bowtie_{in} \sigma_{L< \frac{p}{\text{MIN\_L}(r)}}(r')) \equiv \sigma_{L<p}(\sigma_{L< \frac{p}{\text{MIN\_L}(r')}}(r) \bowtie_{in} r')$;
3. $\sigma_{U\leq p}(r \bowtie_{in} r') \equiv \sigma_{U\leq p}(r \bowtie_{in} \sigma_{U\leq \frac{p}{\text{MIN\_U}(r)}}(r')) \equiv$
   $\sigma_{U\leq p}(\sigma_{U\leq \frac{p}{\text{MIN\_U}(r')}}(r) \bowtie_{in} \sigma_{U\leq \frac{p}{\text{MIN\_U}(r)}}(r')) \equiv \sigma_{U\leq p}(\sigma_{U\leq \frac{p}{\text{MIN\_U}(r')}}(r) \bowtie_{in} r')$;
4. $\sigma_{U<p}(r \bowtie_{in} r') \equiv \sigma_{U<p}(r \bowtie_{in} \sigma_{U< \frac{p}{\text{MIN\_U}(r)}}(r')) \equiv$
   $\sigma_{U<p}(\sigma_{U< \frac{p}{\text{MIN\_U}(r')}}(r) \bowtie_{in} \sigma_{U< \frac{p}{\text{MIN\_U}(r)}}(r')) \equiv \sigma_{U<p}(\sigma_{U< \frac{p}{\text{MIN\_U}(r')}}(r) \bowtie_{in} r')$.

We see from the above results that with some work, we can push probabilistic selections into cartesian products and joins. For example, Theorem 11 and Theorem 12 show that for certain kinds of probabilistics selection conditions, we can easily push selections into Cartesian Product and join. Theorem 13 shows that as long as we maintain MIN_L(r) MIN_U(r) and for each tp-relation r, then we can also push probabilistic selections in Cartesian Product and join when independence is the PCS used. Most relational databases routinely maintain minima and maxima of different columns for query optimization, so this overhead seems acceptable.

4.4 Pushing Projection Through Other Operations

In this section, we present results showing when and how projection can be pushed into TP-algebra operations. Our first result shows that projection can be pushed inside a compaction.

Theorem 14 (Pushing projection into compaction) $\pi_{\mathcal{F}}(\kappa_{\chi}(r)) \equiv \kappa_{\chi}(\pi_{\mathcal{F}}(r))$.

As we have already see before, compaction is part of union and intersection. The above result provides hope that projection can also be pushed through the set operations. This turns out to be true as shown in the following theorem.

Theorem 15 (Pushing projection into set operations) If r and r' have the same schema,

1. $\pi_{\mathcal{F}}(r \cap_{\chi} r') \equiv \pi_{\mathcal{F}}(r) \cap_{\chi} \pi_{\mathcal{F}}(r')$.
2. $\pi_{\mathcal{F}}(r \cup_{\chi} r') \equiv \pi_{\mathcal{F}}(r) \cup_{\chi} \pi_{\mathcal{F}}(r')$.
3. $\pi_{\mathcal{F}}(r - r') \equiv \pi_{\mathcal{F}}(r) - \pi_{\mathcal{F}}(r')$.

The following results indicate that projections can be pushed into Cartesian products and join.

Theorem 16 (Pushing projection into Cartesian product and join) Suppose $\pi_{\mathcal{F}}(r)$ and $\pi_{\mathcal{F}'}(r')$ are TPA-queries. Also, let $\mathcal{F}, \mathcal{F}'$ denote the attribute list that is obtained by concatenating $\mathcal{F}$ with $\mathcal{F}'$ and removing duplicate attribute names. Then

1. $\pi_{\mathcal{F},\mathcal{F}'}(r \times_{\alpha} r') \equiv \pi_{\mathcal{F}}(r) \times_{\alpha} \pi_{\mathcal{F}'}(r')$.
2. $\pi_{\mathcal{F},\mathcal{F}'}(r \bowtie_{\alpha} r') \equiv \pi_{\mathcal{F}}(r) \bowtie_{\alpha} \pi_{\mathcal{F}'}(r')$.

5 Costing TP-Queries

In order to efficiently execute a query in the TP-algebra, we must first have a "cost model" associated with the algebra. Such a cost model has two parts. The first part specifies what statistics to maintain about a TP-relation — such statistics includes information such as cardinality information, distributions of attribute values and so on. The second part deals with the physical costs of executing the operation, which depends upon the implementation. We first discuss the former in section 5.1 and the latter in section 5.3. Due to space restrictions, in this paper, we only consider the Selection, Projection, Cartesian Product and Join operators in the TP-algebra.

5.1 Statistics for TP-Databases

For each TP-relation (base relation or otherwise), we maintain a set of statistics summarized in Table 2. Given a tp-relation r, let $times(r) = \{t \mid$ there exists a tp-tuple tp in r and a tp-case (C, D, L, U, δ) in tp such that $t \in sol(C)\}$. Then MaxC_1$(r) = \max(times(r))$ and MinC_2$(r) = \min(times(r))$.

Statistics	**Description**
CARD	Number of TP-tuples in the TP-relation
AVG_TIME_PTS	Average number of solutions of C constraint in a TP-tuple
MAX_TP	The latest time point in the TP-relation
MIN_TP	The earliest time point in the TP-relation
MaxC_1	See below.
MinC_2	See below.
DOMAIN_MIN(*time unit*)	The smallest value *time unit* can take in calendar τ
DOMAIN_MAX(*time unit*)	The largest value *time unit* can take in calendar τ
AVG_L(U)	Average lower (upper) bound probability in the TP-relation
MIN_L(U)	Minimum lower (upper) bound probability value in the TP-relation
MAX_L(U)	Maximum lower (upper) bound probability value in the TP-relation

Table 2: Statistics for TP-relations.

We now address the following problem. Given a TP-relation r whose statistical variables (cf. Table 2) are known, and given that some TPA operation is executed on this table, how do we *estimate* the values of these statistical variables for the output TP-relation? Though of course this value can be correctly computed by answering the query, we would like to estimate these values *without* answering the query. As in the case of relational database implementations, these *estimates* need to be quickly computable and reasonably (rather than totally !) correct. This allows fast evaluation of many possible different query plans for executing the query, so that we can pick the best. Clearly, for data selects we can use the well-known classical techniques [11]. Hence, we ignore this operation and deal instead with operations new to TPA.

5.1.1 Temporal Selection

Due to space reasons we cannot show how to estimate all these parameters. We describe below, methods to estimate $card(\sigma_{\mathcal{C}}(r))$ and $avg\_time\_pts((\sigma_{\mathcal{C}}(r))$.

Estimating cardinality: $card(\sigma_{\mathcal{C}}(r))$ may be written as $card(r) \times sel(\mathcal{C})$ where $sel(\mathcal{C})$ denotes the *selectivity* of the selection condition $\mathcal{C}$. We can estimate $sel(\mathcal{C})$ by induction on the structure of $\mathcal{C}$.

• $\boldsymbol{tu < value}$: The probability that an arbitrary time point satisfies the constraint $tu \geq value$ is $\left(\frac{DOMAIN\_MAX(tu)-value+1}{DOMAIN\_MAX(tu)-DOMAIN\_MIN(tu)+1}\right)$. For a TP-case in r, the probability that all time points in the solution of that TP-case's C constraint are greater than or equal to $value$ may be estimated by $\left(\frac{DOMAIN\_MAX(tu)-value+1}{DOMAIN\_MAX(tu)-DOMAIN\_MIN(tu)+1}\right)^{AVG\_TIME\_PTS(r)}$. Hence, the probability that at least one such solution of $\mathcal{C}$ will satisfy $tu < value$ is
$1-\left(\frac{DOMAIN\_MAX(tu)-value+1}{DOMAIN\_MAX(tu)-DOMAIN\_MIN(tu)+1}\right)^{AVG\_TIME\_PTS(r)}$, which describes $sel(\mathcal{C})$.

• $\boldsymbol{tu > value}$: The analysis is exactly analogous to the above and the estimate of the selectivity of $\mathcal{C}$ is:

$$sel(\mathcal{C}) = 1 - \left(\frac{value - DOMAIN\_MIN(tu)+1}{DOMAIN\_MAX(tu) - DOMAIN\_MIN(tu)+1}\right)^{AVG\_TIME\_PTS(r)}.$$

• $\boldsymbol{tu \neq value}$: By an analysis similar to that above, we see that here, the estimate of the selectivity of $\mathcal{C}$ is:

$$sel(\mathcal{C}) = 1 - \left(\frac{1}{DOMAIN\_MAX(tu) - DOMAIN\_MIN(tu)+1}\right)^{AVG\_TIME\_PTS(r)}.$$

• $\boldsymbol{tu = value}$: In this case, we simply subtract the selectivity of $\boldsymbol{tu \neq value}$ from 1.

• $\boldsymbol{t_1 \sim t_2}$: On the average, a TP-case has AVG_TIME_PTS solutions to its C constraint. The probability that an arbitrary time point will be a solution of this constraint is $\frac{t_2-t_1+1}{MAX\_TP(r)-MIN\_TP(r)+1}$. The probability that an arbitrary time point will not be a solution of this constraint is
$1-\frac{t_2-t_1+1}{MAX\_TP(r)-MIN\_TP(r)+1}$. Therefore, the probability that no time point of an arbitrary TP-case will be a solution of the constraint $t_1 \sim t_2$ is $\left(1-\frac{t_2-t_1+1}{MAX\_TP(r)-MIN\_TP(r)+1}\right)^{AVG\_TIME\_PTS(r)}$. Hence, the total number of TP-tuples that will not be returned by this operation can be estimated as $CARD^{in} \times \left(1-\frac{t_2-t_1+1}{MAX\_TP(r)-MIN\_TP(r)+1}\right)^{AVG\_TIME\_PTS(r)}$. Then, the expected selectivity is:

$$sel(t1 \sim t2) = 1 - \left(1 - \frac{t_2 - t_1 + 1}{MAX\_TP(r) - MIN\_TP(r)+1}\right)^{AVG\_TIME\_PTS(r)} \qquad \textbf{(Formula TS1)}$$

We can also compute the selectivity of this constraint in another way. Here, we try to estimate the probability that t1 $\sim$ t2 overlaps with any temporal constraint C of the form $t_i \sim t_j$ in the TP-relation. These two intervals may overlap if either (i) $F_1 = t_i < t1$ and $t_j > t1$ (i.e. $t1$ is between t_i and t_j), or (ii) $F_2 = t_i > t1$ and $t_i < t2$ (i.e. t_i is between $t1$ and $t2$). We can compute the selectivities of these conditions as follows:

$sel(t_i < t1) = \begin{cases} 1 & MaxC\_1(r) < t1 \\ 0 & MIN\_TP(r) > t1 \\ \frac{t1 - MIN\_TP(r)}{MaxC\_1(r) - MINTP(r) + 1} & \text{otherwise} \end{cases}$	$sel(t_j > t1) = \begin{cases} 1 & MinC\_2(r) > t1 \\ 0 & MAX\_TP(r) < t1 \\ \frac{MAX\_TP(r) - t1}{MAX\_TP(r) - MinC\_2(r) + 1} & \text{otherwise} \end{cases}$
$sel(t_i > t1) = \begin{cases} 1 & MIN\_TP(r) > t1 \\ 0 & MaxC\_1(r) < t1 \\ \frac{MaxC\_1(r) - t1}{MaxC\_1(r) - MIN\_TP(r) + 1} & \text{otherwise} \end{cases}$	$sel(t_i < t2) = \begin{cases} 1 & MaxC\_1(r) < t2 \\ 0 & MIN\_TP(r) > t2 \\ \frac{t2 - MIN\_TP(r)}{MaxC\_1(r) - MIN\_TP(r) + 1} & \text{otherwise} \end{cases}$

Hence, $sel(F_1) = sel(t_i < t1) * sel(t_j > t1)$ and $sel(F_2) = sel(t_i > t1) * sel(t_i < t2)$. Hence, the selectivity of $(t1 \sim t2)$ will be

$$sel(t1 \sim t2) = sel(F_1) + sel(F_2) - sel(F_1) \times sel(F_2) \qquad \textbf{(Formula TS2)}$$

Later, in Section 6.0.3, we will run experiments to determine which of these two estimates is better.
Non atomic temporal constraints: Any non-atomic temporal constraint can be written purely in terms of $\neg$ and $\wedge$. The selectivity of $\sigma_{\neg\mathcal{C}}(r)$ is equal to 1 minus the selectivity of $\sigma_{\mathcal{C}}(r)$. The selectivity of $\sigma_{\mathcal{C}_1 \wedge \mathcal{C}_2}$ may be obtained by first estimating the selectivity of $r' = \sigma_{\mathcal{C}_2}(r)$ and then estimating $\sigma_{\mathcal{C}_1}(r')$.
Estimating AVG_TIME_PTS : Table 3 shows how we may estimate the average number of time points associated with the C-constraints in a TP-tuple. For space reasons, instead of explaining all the entries, we explain only the first one (which also happens to be the toughest case). The probability that one of these time points has time unit $tu = value$ is $\frac{1}{DOMAIN\_MAX(tu) - DOMAIN\_MIN(tu) + 1}$. As the original TP-relation has $AVG\_TIME\_PTS(r)$ time points in it per TP-tuple, we may therefore assume that the output relation has $\frac{AVG\_TIME\_PTS(r)}{DOMAIN\_MAX(tu) - DOMAIN\_MIN(tu) + 1}$ time points in it per TP-tuple.

5.2 Probabilistic Selection

Our methods to estimate the selectivity of a probabilistic selection condition use the following simple (but useful) result.

Proposition 2 *Given a value* $p \in (0, 1]$ *and a TP-tuple* $tp = (d, \Gamma)$, $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$, $\gamma_i = \langle C_i, D_i, L_i, U_i, \delta_i \rangle$, *there are at most* $\frac{1}{p}$ *time points* $t \in sol(\cup_{i=1}^{n} C_i)$ *such that* $\delta_j(t, D_j) \cdot L_j \geq p$, *where* $t \in sol(C_j)$.

Proof. Let $T_p^* = \{t \in sol(\cup_{i=1}^{n} C_i) | \delta_j(t, D_j) \cdot L_j \geq p, t \in sol(C_j)\}$. Let $s = |T_p^*|$. If $s > \frac{1}{p}$ then $\sum_{t \in T_p^*} \delta_j(t, D_j) \cdot L_j \geq s \cdot p > \frac{1}{p} \cdot p = 1$. As we know that $T_p^* \subseteq \cup_{i=1}^{n} C_i$, $\sum_{t \in T_p^*} \delta_j(t, D_j) \cdot L_j \leq \sum_{i=1}^{n} L_i \leq 1$ which yields a contradiction. $\diamond$

Condition	AVG_TIME_PTS
$tu = value$	$\max(1, \frac{AVG\_TIME\_PTS(r)}{DOMAIN\_MAX(tu) - DOMAIN\_MIN(tu)+1})$.
$tu \neq value$	$AVG\_TIME\_PTS(r) - Max(1, \frac{AVG\_TIME\_PTS(r)}{DOMAIN\_MAX(tu) - DOMAIN\_MIN(tu)+1})$.
$tu > value$	$\frac{AVG\_TIME\_PTS(r) \cdot (DOMAIN\_MAX(tu) - value)}{DOMAIN\_MAX(tu) - DOMAIN\_MIN(tu)+1}$
$tu < value$	$\frac{AVG\_TIME\_PTS(r) \cdot (value - DOMAIN\_MIN(tu))}{DOMAIN\_MAX(tu) - DOMAIN\_MIN(tu)+1}$
tp1 $\sim$ tp2	$AVG\_TIME\_PTS(r) \cdot \frac{tp2 - tp1 + 1}{MAX\_TP - MIN\_TP + 1}$

Table 3: Formulas to compute AVG_TIME_PTS

The above result says that as the probabilities assigned by a PDF add up to 1, at most $\frac{1}{p}$ time points can be assigned a probability greater than or equal to p. We now estimate the selectivity of atomic probabilistic selection conditions, one by one.

We first note that if the value of *prob* in the selection condition is not between $MIN\_L(r)$ ($MIN\_U(r)$) and $MAX\_L(r)$ ($MAX\_U(r)$) for lower(upper) bound conditions, then the selectivity will always be either 0 or 1 depending on the condition. For example if $prob > MAX\_L(r)$ and the condition is $L > prob$ then the result cardinality will be 0 as *no* time point in the TP-relation will satisfy this condition. The table below contains the selectivities of probabilistic selects when *prob* is outside the $[MIN\_L, MAX\_L]$ ($[MIN\_U, MAX\_U]$) interval.

Condition	$prob > MAX\_L(U)$	$prob < MIN\_L(U)$
$L(U) > prob$	0	1
$L(U) < prob$	1	0
$L(U) = prob$	0	0
$L(U) \neq prob$	1	1

We now provide cardinality estimates for probabilistic selects when *prob* is in the above intervals. We propose two sets of cardinality estimations, stemming from two somewhat different approaches. Table 4 summarizes the formulas for the two sets. In this table $PROB_{>}$ denotes the following expression:

$$PROB_{>} = \min\left(1, \frac{1}{prob \times (MAX\_TP(r) - MIN\_TP(r) + 1)}\right).$$

Some explanations about the intuition behind the formulas for both sets are in order.

Set A: We explain the computations behind the **Set A** formulas on the example of atomic probabilistic condition $\boldsymbol{L > prob}$.

Consider an arbitrary TP-tuple in the input relation r. This tuple satisfies the condition $L > prob$ if at least one time point in the solution of one of its C-constraints has the lower bound of probability

Cond. $\mathcal{C}$	**Set A:** $sel(\mathcal{C})$	**Set B:** $sel(\mathcal{C})$
$L = prob$		$\min\left(\frac{1}{prob \times AVG\_TIME\_PTS(r)}, \epsilon\right)$
$L \neq prob$		$1 - \min\left(\frac{1}{prob \times AVG\_TIME\_PTS(r)}, \epsilon\right)$
$L > prob$	$1-(1-PROB_>)^{AVG\_TIME\_PTS(r)}$	$\begin{cases} 1-\epsilon & AVG\_L(r) \geq prob \\ \min(\frac{1}{prob \times AVG\_TIME\_PTS(r)}, (1-prob)) & \text{otherwise} \end{cases}$
$L < prob$	$1-(PROB_>)^{AVG\_TIME\_PTS(r)}$	$\begin{cases} 1 & \text{prob} > 0.5 \text{ and } AVG\_TIME\_PTS(r) \neq 1 \\ 1-\epsilon & \text{prob} > AVG\_L(r) \\ 1-(\frac{AVG\_L(r)-prob}{AVG\_L(r)})^{(AVG\_TIME\_PTS(r)/2)} & \text{otherwise} \end{cases}$
$U = prob$		ϵ
$U \neq prob$		$1-\epsilon$
$U > prob$	$1-(1-PROB_>)^{AVG\_TIME\_PTS(r)}$	$\begin{cases} 1-\epsilon & \text{AVG\_U(r)} > \text{prob} \\ \min(\frac{AVG\_U(r)}{prob}, (1-prob)) & \text{otherwise} \end{cases}$
$U < prob$	$1-(PROB_>)^{AVG\_TIME\_PTS(r)}$	$\begin{cases} 1-\epsilon & \text{prob} > AVG\_U(r) \\ 1-(\frac{AVG\_U(r)-prob}{AVG\_U(r)})^{(AVG\_TIME\_PTS(r)/2)} & \text{otherwise} \end{cases}$

Table 4: Two sets of selectivity estimates for atomic probabilistic selection conditions.

$L > prob$. The probability that an arbitrary time point t satisfies the $L > prob$ constraint can be bounded above by $PROB_> = \min\left(1, \frac{1}{prob \times (MAX\_TP(r) - MIN\_TP(r)+1)}\right)$. Hence, the probability that t does not satisfy this constraint is $1 - \frac{1}{prob \times (MAX\_TP(r)-MIN\_TP(r)+1)})^{AVG\_TIME\_PTS(r)}$ and therefore the probability that a tuple has a time point in its TP-case statement satisfying $L > prob$ is $1-(1-\min(1, \frac{1}{prob \times (MAX\_TP(r)-MIN\_TP(r)+1)}))^{AVG\_TIME\_PTS(r)}$. Hence, the selectivity of $\sigma_{L>prob}(r)$ is given by

$$1-(1-\min(1, \frac{1}{prob \times (MAX\_TP(r) - MIN\_TP(r)+1)}))^{AVG\_TIME\_PTS(r)}.$$

The same reasoning applies to the all remaining atomic constraints on lower and upper bounds featured in Table 4.

<u>Set B:</u> The intuition for different atomic conditions is somewhat different.

L = prob. By Proposition 2 there can be at most $\frac{1}{prob}$ time points with probability equal to *prob* in any TP-tuple. Therefore, when the number of time points in a TP-tuple is large, we expect that the chances of any time point to have a fairly large lower bound of the probability interval are fairly small. This inequality between small and large values of *prob* is represented by the expression $\frac{1}{prob \times AVG\_TIME\_PTS(r)}$.

On the other hand, the chances of finding a time point with any particular *small* lower bound of the probability interval should be approximately the same. We represent that, by assuming that there exists a small constant ϵ that serves as an upper bound on the probability to encounter a time point satisfying $L = prob$. As there is a finite number of time points in the TP-tuple and, potentially a continuum of values *prob* can take, ϵ should be a relatively small number.

L > prob. Whenever *prob* is less than or equal to the average lower bound probability value for a single time point, we are all but guaranteed the existence of at least on time point with lower bound larger than *prob*. Otherwise, if *prob* is very close to 1, then there is very little chance of finding a time point with a larger lower bound. $1 - prob$ describes this value, in a min computation this will "win" whenever *prob* is almost 1. For the values of *prob* between the average lower bound and 1, the chance to find a large probability value is in inverse proportion to the number of time points in a TP-tuple and to the value of *prob* (c.f. Proposition 2).

***L* < *prob*.** Whenever $prob > 0.5$ and there is more than one time point in a TP-tuple, we are guaranteed to have a time point having a probability less than 0.5 and hence less than *prob*. If *prob* is greater than the average probability then similar to the case above, the existence of a time point with a lower bound smaller than *prob* is almost guaranteed, because there have to be time points with probability less than the average probability. In the last entry in the above formula, we assume that the expected number of time points with lower bounds greater than $AVG\_L(r)$ os $AVG\_TIME\_PTS(r)/2$. $\frac{AVG\_L(r)-prob}{AVG\_L(r)}$ provides our estimate of probability that a given time point is in the interval between *prob* and the average lower bound. Such points do not satisfy our requirements. So, by computing the probability that half of all time points are in this range (assuming that the other half is above the average), and subtracting it from 1, we obtain a probability estimate for the existence of at least one time point with lower bound for probability under *prob*.

The estimates for the remaining conditions use similar intuitions (correcting for the fact that Proposition 2 does not apply to the upper bounds of probability intervals).

Whether **Set A** or **Set B** estimates are used, it is easy to estimate cardinality of queries such as $\sigma_{L\in[p1,p2]}(r)$ by first estimating the size of $\sigma_{L\geq p1}(r)$, calling the result r' and then estimating $\sigma_{L<p2}(r')$.

Other statistical variables: Table 5 specifies how, given a probabilistic selection condition (atomic), we may use the values of the statistical variables associated with input relation r to estimate the values of AVG_L, AVG_U for the output relation obtained by applying the select.

Condition	AVG_L	AVG_U
$L = prob$	$prob$	$prob + AVG\_DIFF$
$L > prob$	$\max(AVG\_L^{in}, prob) + sel \cdot AVG\_L^{in}$	$AVG\_L^{out} + AVG\_DIFF$
$L < prob$	$\min(AVG\_L^{in}, prob) - sel \cdot AVG\_L^{in}$	$AVG\_L^{out} - AVG\_DIFF$
$U = prob$	$prob - AVG\_DIFF$	$prob$
$U > prob$	$AVG\_U^{out} + AVG\_DIFF$	$\max(AVG\_U^{in}, p) + sel \cdot AVG\_U^{in}$
$U < prob$	$AVG\_U^{out} - AVG\_DIFF$	$\min(AVG\_U^{in}, p) - sel \cdot AVG\_U^{in}$
$L\ IN\ [p1, p2]$	$\max(p1, (\min(AVG\_L^{in}, p2) - sel(L \leq p2) \cdot AVG\_L^{in})) + (\min(AVG\_L^{in}, p2) - sel(L \leq p2) \cdot AVG\_L^{in}) \cdot sel(L \geq p1)$	$AVG\_L^{out} + AVG\_DIFF$
$U\ IN\ [p1, p2]$	$AVG\_U^{out} + AVG\_DIFF$	$\max(p1, (\min(AVG\_U^{in}, p2) - sel(U \leq p2) \cdot AVG\_U^{in})) + (\min(AVG\_U^{in}, p2) - sel(U \leq p2) \cdot AVG\_U^{in}) \cdot sel(U \geq p_1)$

Table 5: Computations for AVG_L and AVG_U in the result relation

5.2.1 Cartesian Product and Join

Given two arbitrary TP-tuples $tp \in r$ and $tp' \in r'$ ($tp = (d, \Gamma), \Gamma = \{\gamma_1, \ldots, \gamma_n\}), \gamma_i = \langle C_i, D_i, L_i, U_i, \delta_i\rangle$; $tp' = (d', \Gamma'), \Gamma' = \{\gamma'_1, \ldots, \gamma'_m\}), \gamma'_j = \langle C'_j, D'_j, L'_j, U'_j, \delta'_j\rangle)$, we need to determine the probability that there will be a TP-tuple $tp'' \in r \times_\alpha r'$ corresponding to these two tuples. The Cartesian product of tp and tp' will be nonempty if they share at least one time point, i.e., if there exists $1 \leq i \leq n$, $1 \leq j \leq m$, such that $sol(C_i \wedge C'_j) \neq \emptyset$.

Let $C = \bigcup_{i=1}^{n} C_i$ and $C' = \bigcup_{j=1}^{m} C'_j$. We first need to compute the probability that a solution of C is also a solution of C'. Note that if t is *outside* the range of time points found in r' ($[MIN\_TP(r'), MAX\_TP(r')]$) then it is definitely **not** a shared solution. Therefore, we first establish the probability that $t \in [MIN\_TP(r'), MAX\_TP(r')]$. As $t \in sol(C)$, we know that $t \in [MIN\_TP(r), MAX\_TP(r)]$.

Let $TR(r, r') = |[MIN\_TP(r), MAX\_TP(r)] \cap [MIN\_TP(r'), MAX\_TP(r')]|$. Then

$$Pr(t \in [MIN\_TP(r'), MAX\_TP(r')] | t \in [MIN\_TP(r), MAX\_TP(r)]) = \frac{TR(r, r')}{MAX\_TP(r') - MIN\_TP(r') + 1}.$$

Once we establish that t is in the range of r', we need to determine the probability that $t \in sol(C')$. This probability is given by $\frac{AVG\_TIME\_PTS(r')}{MAX\_TP(r') - MIN\_TP(r') + 1}$.

We obtain the desired probability $Pr(t \in sol(C') | t \in sol(C))$ by multiplying the two numbers:

$$sel_1 = Pr(t \in sol(C') | t \in sol(C)) = \frac{TR(r, r')}{MAX\_TP(r') - MIN\_TP(r') + 1} \times \frac{AVG\_TIME\_PTS(r')}{MAX\_TP(r') - MIN\_TP(r') + 1}.$$

By a symmetric argument,

$$sel_2 = Pr(t \in sol(C) | t \in sol(C')) = \frac{TR(r, r')}{MAX\_TP(r) - MIN\_TP(r) + 1} \times \frac{AVG\_TIME\_PTS(r)}{MAX\_TP(r) - MIN\_TP(r) + 1}.$$

Hence, the cardinality of the Cartesian product can be estimated as

$$CARD(r) \times CARD(r') \times \max(sel_1, sel_2).$$

As the join operation in TP-algebra is merely a Cartesian product followed by a selection, the cardinality of join is given by:

$$CARD(r) \times CARD(r') \times \max(sel_1, sel_2) \times sel(\mathcal{JC}),$$

where $\mathcal{JC}$ is the join condition whose selectivity is computed in the same way as in the relational case [11].

5.3 Physical costs

In order to specify the physical costs of executing these operations, we must first provide a brief description of the implementation of TP-databases.

5.3.1 Implementation Overview

We have significantly extended the implementation of TP-databases outlined in [5] by (i) adding a probabilistic table index, and (ii) developing a cost model that uses the physical implementation of the algebra operations using a relational implementation and the probabilistic table index as well as (well known) segment tree indexes for temporal data, (iii) developing a set of rewrite rules based

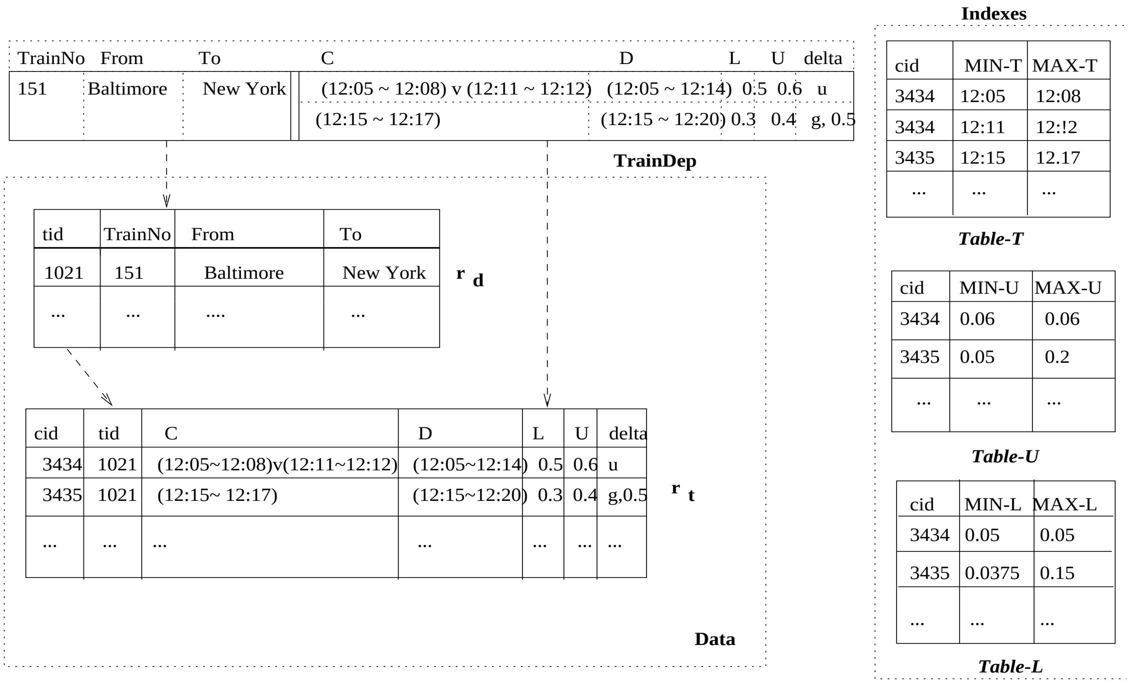


Figure 1: Storing TP-tuples in Paradox: r_d and r_t relations and *Table-L*, *Table-U* and *Table-T* indexes.

on the query equivalences we have derived in Section 4 and (iv) building a query optimizer for TP-databases using the above components and the Cascades optimizer framework [12]. In this section, we briefly describe the implementation.

A TP-relation consists of three parts – *data*, *temporal* and *probabilistic*. Each TP-relation r over relational schema $(A_1, \ldots, A_k)$ is stored as two Paradox tables: r_d over schema (`TId`, A_1, …, A_k), and r_t over schema (`CId`, `TId`, C, D, L, U, Delta). Here, `TId` is the *TP-tuple id* which is used to join r_d and r_t when a user wants to view r, and `CId` is the *TP-case id* which is used to uniquely identify a TP-case. We create a primary key index on `TId` for r_d, and on `CId` for r_t. To perform a query on r, we use the Borland Database Engine (BDE) to perform relational queries on Paradox tables r_d and r_t. Moreover, the TP-relation r'' for the result of a query is materialized as two Paradox tables (r''_d and r''_t).

Probabilistic table index: To index probabilistic information, we first create index tables *Table-*L_r and *Table-*U_r for each TP-relation r. We explain the structure of *Table-*L_r below - the other table is built in a similar way. *Table-*L_r has the schema (cid, MIN-L, MAX-L). For each tuple $(cid, C, D, L, U, \delta) \in r_t$ we store in *Table-*L_r, its cid and the values

$$\text{MIN-L} = \min_{t \in sol(C)} (L \cdot \delta(D, t))$$
$$\text{MAX-L} = \max_{t \in sol(C)} (L \cdot \delta(D, t))$$

MIN-U and MAX-U values for each TP-case in *Table-*U_r may be stored in a similar way.

Each probabilistic query allowed in TPA can be expressed as a combination of $L \in \mathsf{Interval}$ and $U \in \mathsf{Interval}$ (where Interval can either be open or closed on either side). For each such request $L \in [a, b]$, *Table-*L_r is scanned and the set of cids is determined such that $[a, b] \cap [\mathsf{MIN\text{-}L}, \mathsf{MAX\text{-}L}] \neq \emptyset$. This will be the list of *candidate* TP-cases. We are guaranteed that the answer to $L \in [a, b]$ is in a subset of these candidate TP-cases. Each then needs to be retrieved and checked.

Temporal index: For indexing temporal data we use three different data structures. Two of them are variations of *segment trees* [28] and the third is a variation of the index table structure used to index probabilistic information. As segment trees are well studied, we do not discuss them further here. Future work may involve extending sophisticated temporal index structures such as those developed by Tsotras[18, 34] to handle probabilistic temporal data.

The tabular index structure *Table-*T_r for the temporal data contains three fields: cid, the TP-case id in r_t and MAX-T and MIN-T – the maximal and minimal time point for each contiguous interval described in TP-case with id cid. Unlike the case with *Table-*L_r and *Table-*U_r, where cid was a unique foreign key, cid will no longer be a unique key for *Table-*T_r. On the other hand, the result of performing a select on a temporal condition using *Table-*T_r will be the exact set of TP-case ids matching the query (not a set of candidate TP-cases).

Figure 1 shows how temporal probabilistic data of Example 1 is stored and indexed in the underlying Paradox database.

5.3.2 Physical Cost Model

The TP-database implementation builds on top of a fixed set of query templates that are used to access the underlying Paradox tables in which TP-relations are stored. Each TP-algebra operator is encoded via a C++ program that builds on such top of these templates. To model the physical cost of such operators, we must therefore: (i) model the costs of the templates, and (ii) use the template models to model the costs of the C++ programs' encoding the different TP-algebra operators. derive a formula that estimates the cost of how long such C++ programs need to run. Solely using a standard cost model for relational databases is not enough.

Query Templates: The set of Paradox query templates and their associated cost formulas are provided in Table 6. We use simple calibration and regression analysis techniques, such as those in [35] to compute the costs of these template queries.

The notation used in Table 6 requires further explanation: A_d and A_t denote the data and TP-case tables for an input TP-relation A. C_d and C_t denote the data and TP-case tables for the resulting TP-relation. For binary operations like join and Cartesian product, B_d and B_t denote the data and TP-case tables of the second operand. Two intermediate tables $TmpF$ and $TmpJ$, which will be discussed below, are used for Cartesian product and join operations. $tableIndex$ denotes a temporal or probabilistic index which is maintained as a Paradox table. Finally, we use $[r]$ to denote the schema of relation r.

The cost formulas in Table 6 were obtained by running several queries, trying many different cost formulas, and using standard regression analysis and calibration methods to fit the best curves (and hence obtain the best formulas). The cost formulas for query templates 1-8 are linear combinations of input and output table cardinalities — this is to be expected as they involve a simple linear

#	Query Template	Cost Formula
1	$C_d \leftarrow C_d \cup \sigma_{\mathcal{C}}(A_d)$	$a \cdot \lvert A_d\rvert + b \cdot \lvert C_d\rvert + c$
2	$C_t \leftarrow C_t \cup \pi_{[A_t]}(\sigma_{(A_t.\mathrm{TId}=C_d.\mathrm{TId})}(A_t \times C_d))$	$a \cdot \lvert A_t\rvert + b \cdot \lvert C_d\rvert + c$
3	A_t	$a \cdot \lvert A_t\rvert + b$
4	$C_d \leftarrow C_d \cup \pi_{[A_d]}(\sigma_{(A_d.\mathrm{TId}=C_t.\mathrm{TId})}(A_d \times C_t))$	$a \cdot \lvert A_d\rvert + b \cdot \lvert C_t\rvert + c$
5	$\sigma_{(\mathrm{CId}\in S)}(A_t)$	$a \cdot \lvert A_t\rvert + b \cdot \lvert S\rvert + c$
6	$T \leftarrow \sigma_{\mathcal{C}}(tableIndex)$; $\pi_{[A_t]}(\sigma_{(A_t.\mathrm{CId}=T.\mathrm{CId})}(A_t \times T))$	$a \cdot \lvert A_t\rvert + b \cdot sel(\mathcal{C}) \cdot \lvert tableIndex\rvert + c$
7	$C_d \leftarrow C_d \cup \pi_{\mathcal{F}}(A_d)$	$a \cdot \lvert A_d\rvert + b$
8	$C_t \leftarrow C_t \cup A_t$	$a \cdot \lvert A_t\rvert + b$
9	$A_t \times B_t$	$a \cdot \max(\lvert A_t\rvert, \lvert B_t\rvert) + b \cdot \lvert A_t\rvert \cdot \lvert B_t\rvert + c$
10	$T \leftarrow \sigma_{(A_d.\mathrm{TId}=\mathrm{TmpA}) \wedge (B_d.\mathrm{TId}=\mathrm{TmpB})}(A_d \times B_d \times TmpF)$; $C_d \leftarrow C_d \cup \pi_{\mathrm{TmpC},[A_d],[B_d]}(T)$	$a \cdot \lvert TmpF\rvert + b$
11	$TmpJ \leftarrow TmpJ \cup \pi_{A_d.\mathrm{TId},B_d.\mathrm{TId}}(\sigma_{\mathcal{C}}(A_d \times B_d))$	$a \cdot \min(\lvert A_d\rvert, \lvert B_d\rvert) + b \cdot \lvert TmpF\rvert + c$
12	$T \leftarrow \sigma_{(A_t.\mathrm{TId}=\mathrm{TmpA}) \wedge (B_t.\mathrm{TId}=\mathrm{TmpB})}(A_t \times B_t \times TmpJ)$; $\pi_{[A_t],[B_t]}(T)$	$\mathrm{cost}(A_t \bowtie TmpJ) +$ $\mathrm{cost}(B_t \bowtie TmpJ) + \epsilon$
13	$A'_d \leftarrow \pi_{\mathcal{F}}(A_d)$; $B'_d \leftarrow \pi_{\mathcal{F}}(B_d)$; $T \leftarrow \sigma_{(A'_d.\mathrm{TId}=\mathrm{TmpA}) \wedge (B'_d.\mathrm{TId}=\mathrm{TmpB})}(A'_d \times B'_d \times TmpF)$; $C_d \leftarrow C_d \cup \pi_{\mathrm{TmpC},[A'_d],[B'_d]}(T)$	$a \cdot \lvert TmpF\rvert + b$

Table 6: Query Templates Used by Operators in the TP-algebra and Their Cost Formulas

relational operation. Likewise, *Query 9* is a Cartesian product operation, and its cost formula involves the products of the TP-relations involved.
In the case of *Queries 10 and 13*, we observed that the cost largely depends on the cardinality of *TmpF*. Recall that `TId` is a primary key and there exists an index on `TId` in the data tables. The join conditions in queries 10 and 13 are on `TId`. Hence, *TmpF* needs to be read once, and all tuples satisfying the join are retrieved via the index on A_d, and similarly for B_d.
Query 11 is a join between two data tables. The join condition is on a data field and there is no index on data fields. One would assume that Paradox uses either a nested loop or a merge join. After extensive calibration, the best result seems to be the expression shown in Table 6.
Finally, *Query 12* involves a two way join. We were unable to fit a satisfactory curve to match the behavior of this query. As a consequence, we tried to model the cost of this query as two successive join operations, and this led to the cost formula shown.
TP-Algebra Operations: We are now ready to use the costs of query templates in order to derive cost formulas for TP-algebra operations. Each query begins by creating the C_d and C_t tables which will hold the results. To compute costs of TP-algebra operations, we executed each operation several times to determine its average running times. Our cost formulas focus on the cost of retrieving and storing data in the Paradox tables. We use CREATE_COST(T) to denote the cost of creating Paradox table T, GET_NEXT_COST to denote the cost of processing one Paradox tuple via a Borland Data Engine (BDE) cursor[1], and INSERT_COST(T) denotes the cost of inserting one tuple into Paradox table T. For brevity, let CREATE_COST denote CREATE_COST(C_d) +

[1] We assume all readers are familiar with standard database concepts like cursors[11].

CREATE_COST(C_t).

Data Select: As this operation corresponds to a relational select, the data condition $\mathcal{C}$ is simply passed to the BDE. Thus, C_d is computed by *query 1*, and C_t by *query 2*. The cost formula is

$$cost \quad = CREATE\_COST + cost(query1) + cost(query2).$$

Temporal Select (without index): A temporal condition $\mathcal{C}$ is evaluated using special purpose code since there is no equivalent notion in a relational database. Consider a boolean combination of temporal intervals of the form $(t_i \sim t_j)$. As there is no index, all TP-cases in A_t need to be retrieved and checked. Thus we open a cursor on the result of *query 3*, read one TP-case γ_i at a time, compute $C = (\gamma_i.C \wedge \mathcal{C})$, and insert a new TP-case which incorporates C into C_t if sol$(C) \neq \emptyset$. Once the computation of C_t is finished, the appropriate tuples are inserted into C_d by executing *query 4*. The cost formula is

$$\begin{aligned} cost \quad = & CREATE\_COST + cost(query3) + card(A_t) \times GET\_NEXT\_COST + \\ & card(C_t) \times INSERT\_COST(C_t) + cost(query4). \end{aligned}$$

Temporal Select (with segment tree index): In this case, we first retrieve a set S of `CId`s by using the segment tree index. This index filters out irrelevant data by trying to ensure that $(\gamma_i.C \wedge \mathcal{C}) \neq \emptyset$ for each TP-case γ_i whose `CId` is in S. We then execute *query 5* to retrieve the TP-cases referred to in S. We read one TP-case at a time from the result of *query 5*, compute C, and write the result into C_t. Finally, C_d is populated by executing *query 4*. The cost formula is

$$\begin{aligned} cost \quad = & CREATE\_COST + cost(\texttt{segTreeSearch}) + cost(query5) + \\ & \mid S \mid \times GET\_NEXT\_COST + card(C_t) \times INSERT\_COST(C_t) + cost(query4). \end{aligned}$$

Temporal Select (with table index *Table-T_r*): *Table-T_r* is stored as a Paradox table where MIN-T and MAX-T are indexed using standard, Paradox indices. We first execute *query 6* to retrieve from this table all `CId`s of TP-cases in A_t whose C constraints overlap with the selection condition. We then open a cursor on the result of *query 6*, read one tuple at a time, compute C, and save the results in C_t. We finally execute *query 4* to compute C_d. Hence, the cost formula is

$$\begin{aligned} cost \quad = & CREATE\_COST + cost(query6) + sel(\mathcal{C}) \times card(tableIndex) \times \\ & GET\_NEXT\_COST \;\; + card(C_t) \times INSERT\_COST(C_t) + cost(query4). \end{aligned}$$

Probabilistic Select (without index): As relational databases do not support probabilistic selection conditions, we use special purpose code to compute the results of this operation which is similar to the unindexed temporal select operation. The only difference is that C is computed via TP-filter$(\gamma_i, \mathcal{C})$, and hence sol(C) will be the set of time points in $\gamma_i.C$ which satisfy $\mathcal{C}$. The cost formula is the same as the one used for unindexed temporal selections.

Probabilistic Select (with table index): A probabilistic condition $\mathcal{C}$ can be rewritten using one interval (if `op` is not "$\neq$"), or the union of two intervals. For instance, $L > p$ can be rewritten as $L \in (p, 1]$. As a selection condition is a disjunction of intervals, probabilistic select queries are implemented by using a procedure similar to the one used for temporal selects with table indices. In fact, both queries share the same cost formula.

Project: As the TP-algebra only allows projections of data attributes, this operator only projects out fields from A_d and retains all fields in A_t by executing *query 7* and *query 8* respectively. The cost formula is

$$cost \quad = CREATE\_COST + cost(query7) + cost(query8).$$

Cartesian Product: To execute a Cartesian product operation between TP-relations A and B, we first execute *query 9* to retrieve all tuple ids from A_t and B_t. For each unique pair of TP-tuples from A and B that produces a result in the Cartesian product, a new, unique tuple id needs to be created for the resulting tuple. The TP-algebra uses a mapping f which takes tuple ids from A and B as input, and returns a new tuple id for C. This mapping is stored as a Paradox table *TmpF* with the schema (TmpA, TmpB, TmpC). Here, TmpA, TmpB, and TmpC refer to the `TId`s from A, B, and C respectively. The implementation then creates *TmpF*, whose cardinality equals that of C_d. We then open a cursor on the result of *query 9*, and read one tuple at a time. For each tuple in the result of *query 9*, we compute $C = (A_t.C \wedge B_t.C)$ and discard this tuple if $\mathrm{sol}(C) = \emptyset$. Otherwise we generate a unique id for C_t.`CId`, and determine whether or not there is a value for $f(A_t.\texttt{TId}, B_t.\texttt{TId})$. If not, we generate a unique id for C_t.`TId`, and assign this value to $f(A_t.\texttt{TId}, B_t.\texttt{TId})$. In either case, the implementation inserts $\langle C_t.\texttt{CId}, f(A_t.\texttt{TId}, B_t.\texttt{TId}), C, D, L, U, \delta\rangle$ into C_t when these values satisfy Definition 8 of Cartesian product. Note that δ may be a new distribution function that distributes $[L, U]$ among all time points $t \in \mathrm{sol}(D)$ according to the probabilities determined by $\otimes_\alpha$. After all tuples returned by *query 9* are processed, function f is saved. For all a, b, c where $f(a, b) = c$, the code inserts (a, b, c) into *TmpF*. This information is used to populate C_d by executing *query 10*. Finally, we remove *TmpF*. The cost formula is

$$\begin{aligned} cost \quad =& CREATE\_COST + CREATE\_COST(TmpF) + cost(query9) + \\ & card(A_t) \times card(B_t) \times GET\_NEXT\_COST + card(C_t) \times INSERT\_COST(C_t) + \\ & card(TmpF) \times INSERT\_COST(TmpF) + cost(query10). \end{aligned}$$

Join: To execute the join operation, we first create a temporary Paradox table *TmpJ* with the schema (TmpA, TmpB), and populate *TmpJ* by executing *query 11*. *TmpJ* stores the `TId`s of the pairs which satisfy the join condition $\mathcal{C}$. Hence, the cardinality of *TmpJ* is $\mathrm{card}(A_d) \times \mathrm{card}(B_d) \times sel(\mathcal{C})$. The rest of the join code is similar to that of Cartesian product except that *Query 12* is used instead of *Query 9* (to limit the number of TP-cases visited), and *Query 13* is used instead of *Query 10* (as the result requires a projection on $\mathcal{F}$). Furthermore, *TmpJ* can be removed any time after executing *query 12*. The cost formula is

$$\begin{aligned} cost \quad =& CREATE\_COST + CREATE\_COST(TmpJ) + cost(query11) + \\ & CREATE\_COST(TmpF) + cost(query12) + \\ & card(TmpJ) \times GET\_NEXT\_COST + card(C_t) \times INSERT\_COST(C_t) + \\ & card(TmpF) \times INSERT\_COST(TmpF) + cost(query13). \end{aligned}$$

6 Experimental Results

We conducted experiments aimed at (i) evaluating the effectiveness of our selectivity estimates, (ii) evaluating the effectiveness of our rewrite rules, and (iii) evaluating the effectiveness of our TP-optimizer as a whole.

All these experiments use a table creation program (TCP) which generates tables in accordance with various parameters described in Table 7. TCP generates NO_TUPLES number of TP-tuples. The TP-relations generated by TCP contain data fields $f1$ and $f2$. TCP assigns unique consecutive integers to the $f1$ field, and randomly assigns integers in the interval [1, F2_Range] to the $f2$ field. To generate a temporal constraint C of the form $t_1 \sim t_2$, a value is drawn randomly from the range C_R for t_1. The number of time points that are solutions to a TP-case is controlled by drawing value x from the range AVG_TIME_PTS_R and adding it to t_1. Values for L and U are generated randomly by choosing a value from the ranges L_R and U_R respectively. Distributions in tp-cases are restricted to geometric (g), binomial (b), uniform (u) and mix. When a "mix' is chosen, TCP randomly picks one of the g, b or u distributions.

Parameter	**Description**
NO_TUPLES	number of TP-tuples
F2_Range	range for the data attribute f2
C_R	range for the lower bound of the temporal constraint C
AVG_TIME_PTS_R	range for the average number of time points in a TP-tuple
L_R	range for the lower bound probability L
U_R	range for the upper bound probability U
Delta	probability distribution function

Table 7: Parameters of the Table Creation Program

6.0.3 Effectiveness of Selectivity Estimators

We used TCP to create 6 TP-relations with various cardinalities and properties. These are shown below. Each TP-tuple has one case.

TP-rel	**NO_TUPLES**	**F2_Range**	**C_R**	**AVG_TIME_PTS_R**	**L_R**	**U_R**	**Delta**
r1	100	-	1900-2000	5-10	0.8-1.0	0.9-1.0	mix
r2	1000	-	1800-2000	5-15	0.75-0.9	0.95-1.0	mix
r3	10000	-	1800-2010	10-20	0.8-0.95	0.9-1.0	mix
r4	1000	-	1950-2020	5-10	0.75-0.95	0.9-1.0	mix
r5	1000	-	1900-1980	10-20	0.6-0.8	0.9-1.0	mix
r6	10000	-	1850-2020	5-15	0.8-1.0	0.9-1.0	mix

Table 8: TP-relations used in the first set of experiments.

Our experiments focus on the effectiveness of the selectivity estimation formulas for temporal and probabilistic selection conditions as well as join operations. Table 9 contains the temporal and probabilistic selection queries we used in these experiments.

Temporal Selectivity: Figure 2 shows the actual cardinality of the resulting answers, as well as the estimated values provided by **Formulas TS1, TS2**. The leftmost bar in each group in the figure

#	Query
1	$\sigma_{1980\sim1995}(r_2)$
2	$\sigma_{1940\sim1970}(r_2)$
3	$\sigma_{1920\sim1995}(r_2)$
4	$\sigma_{1870\sim1990}(r_2)$
5	$\sigma_{1950\sim1960}(r_4)$
6	$\sigma_{1975\sim1993}(r_4)$
7	$\sigma_{1900\sim1990}(r_4)$
8	$\sigma_{1880\sim2000}(r_4)$
9	$\sigma_{1965\sim1975}(r_5)$
10	$\sigma_{1960\sim2010}(r_5)$
11	$\sigma_{1850\sim1950}(r_5)$
12	$\sigma_{1940\sim1980}(r_5)$

(a) Temporal Selection Queries

#	Query
1	$\sigma_{L>0.4}(r_2)$
2	$\sigma_{L>0.15}(r_5)$
3	$\sigma_{L>0.1}(r_4)$
4	$\sigma_{L>0.05}(r_2)$
5	$\sigma_{L<0.01}(r_4)$
6	$\sigma_{L<0.017}(r_2)$
7	$\sigma_{L<0.07}(r_5)$
8	$\sigma_{L<0.5}(r_4)$
9	$\sigma_{U>0.48}(r_4)$
10	$\sigma_{U>0.25}(r_5)$
11	$\sigma_{U>0.27}(r_2)$
12	$\sigma_{U>0.2}(r_4)$
13	$\sigma_{U<0.01}(r_4)$
14	$\sigma_{U<0.01}(r_5)$
15	$\sigma_{U<0.1}(r_2)$
16	$\sigma_{U<0.3}(r_5)$

(b) Probabilistic Selection Queries

Table 9: Selection Queries for Experiments

shows the cardinality estimates generated by using **Formula TS1**, the middle bar show the estimates generated by **Formula TS2** and the rightmost bar shows the actual result cardinalities. It is easy to see that **Formula TS2** generates better results than **Formula TS1**.

Figures 3, 4, 5 and 6 contain the actual cardinality and the cardinality estimates generated by **Set A** and **Set B** formulas for probabilistic conditions of the form $L > prob$, $L < prob$, $U > prob$ and $U < prob$, respectively. As seen from the figures, **Set B** formulas provide better estimates than **Set A** formulas. For $L > prob$, **Set B** results are query accurate (we surmise that this is because they use Proposition 2). For the same reason, and the fact that the sum of the upper probability values do not have to add up to 1, but add up (on average) to $AVG\_U \times AVG\_TIME\_PTS$, the formula for $U > prob$ also performs very well. The formulas for $L < prob$ and $U < prob$ in **Set B** assume that the number of time points having probability values less than AVG_L (AVG_U) is half of the average time points in a TP-tuple. In fact, we expect that there are more time points with lower bounds of probability intervals between 0 and AVG_L than there are points with the lower bounds of probability intervals between AVG_L and 1, as the sum should add up to 1. Hence, the estimates for $L < prob$ values are not as accurate as the $L > prob$ case. A symmetric argument also applies to the $U < prob$ condition. The cardinality estimate generated by **Set A** for queries 5 and 13 in Table 9(b) were 0, and hence hence they do not appear in Figures 4 and 6.

Join: In order to assess the accuracy of our join size estimators, we created 7 queries involving a two-way join between various TP-relations. These queries and their actual and the estimated cardinalities are shown in Table 10. All queries assume that no prior knowledge about the relationship between the

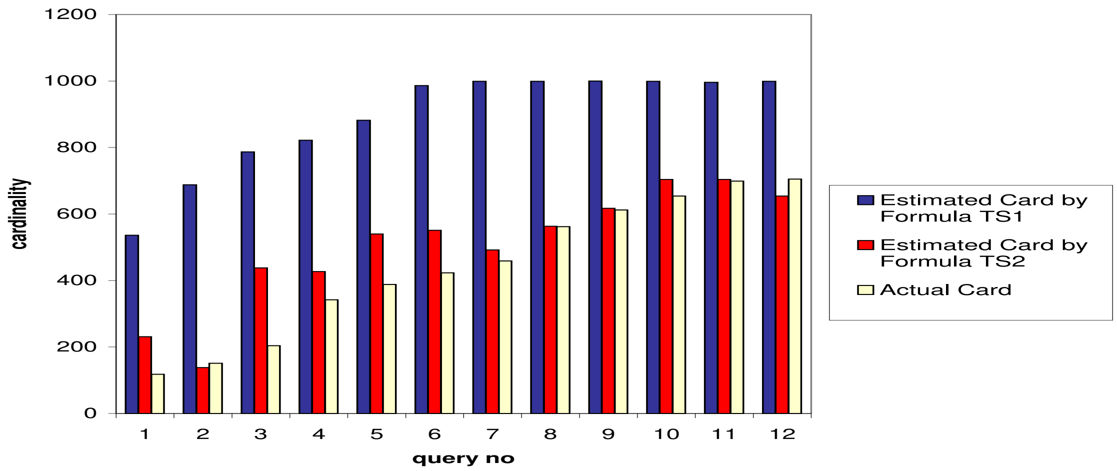


Figure 2: Actual And Estimated Cardinalities for Temporal Selections

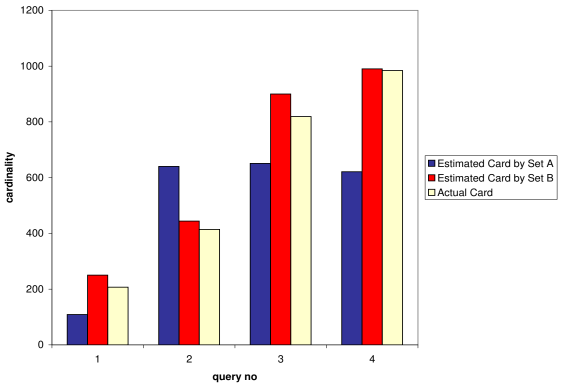


Figure 3: Actual And Estimated Cardinalities for $L > prob$

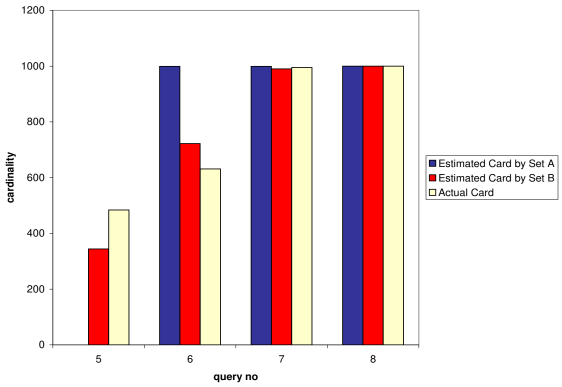


Figure 4: Actual And Estimated Cardinalities for $L < prob$

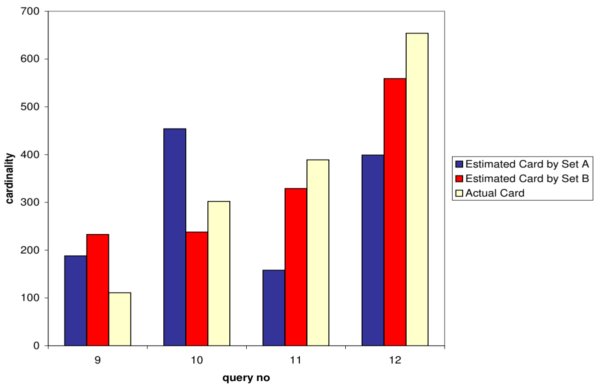


Figure 5: Actual And Estimated Cardinalities for $U > prob$

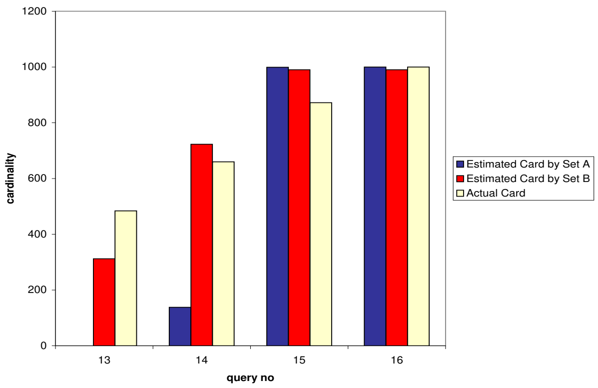


Figure 6: Actual And Estimated Cardinalities for $U < prob$

events represented in base relations exist – an assumption, known as *ignorance* [20]. The conjunction strategy for this can be specified as follows: $[a,b] \otimes_{ig} [c,d] = [\max(0, a+c-1), \min(b,d)]$. The results indicate that the cardinality estimates for the join provide "decent" estimates. The estimated cardinalities seem to be reasonably accurate.

Query No	QUERY	Actual Card	Estimated Card
1	select * from r1, r6 under ig where r1.f1 = r6.f1	7	7
2	select * from r1, r3 under ig where r1.f1 = r3.f1	10	7
3	select * from r2, r4 under ig where r2.f1 = r4.f1	57	79
4	select * from r2, r6 under ig where r2.f1 = r6.f1	94	47
5	select * from r4, r6 under ig where r4.f1 = r6.f1	113	97
6	select * from r2, r3 under ig where r2.f1 = r3.f1	124	64
7	select * from r3, r5 under ig where r3.f1 = r5.f1	169	147

Table 10: Join Queries for Experiments

6.0.4 Effectiveness of Rewrite Rules

In this section, we study the effectiveness of some of the rewrite rules proved in Section 4. Due to space reasons, we are unable to present results on the effectiveness of all the rewrite rules. We focus on the result that temporal and probabilistic selects are commutative (Theorem 4), and that temporal selects can be pushed into joins (Theorem 9).

Commutativity of Temporal and Probabilistic Selects: If T (resp. P) is a temporal (resp. probabilistic) selection condition, then by Theorem 4, we know that $\sigma_P(\sigma_T(r)) = \sigma_T(\sigma_P(r))$. We started by generating TP-relations using the TCP with the following parameters:

TP-rel	NO_TUPLES	F2_Range	C_R	AVG_TIME_PTS_R	L_R	U_R	Delta
r	3880	1-100	2000-2090	1-10	0.1-0.5	0.1-1.0	mix

To vary T, we used the temporal condition $(2000 \sim (2000 + 10 \cdot i))$ for all $i \in [0,9]$. To vary P, we used the probabilistic condition $(L \le 0)$, $(L \le 0.32 \cdot (\frac{1}{2})^i)$ for all $i \in [0,8]$, and $(L \le 1)$. For each pair (T,P), we determined the time (in seconds) to compute $\sigma_P(\sigma_T(r))$ minus the time to compute $\sigma_T(\sigma_P(r))$ when r is not indexed (i.e., when all TP-cases must be examined). If this difference is positive (negative), then it is better to perform the probabilistic (temporal) select first respectively. The results are shown in Figure 7.

In general, if Sel-T and Sel-P are approximately equal, then both orderings are approximately equal. Otherwise, it is better to perform the selection that offers the highest selectivity first. In other words, even though it takes a lot of time to compute TP-filter(γ_i, P) used in probabilistic as compared to solving the constraint $\gamma_i.C \wedge T$ in the case of temporal selects, this difference turns out to be negligible. Intuitively, this occurs because testing γ_i is mostly a CPU-intensive operation while writing out satisfying TP-cases is a memory-intensive operation, and hence is more expensive.

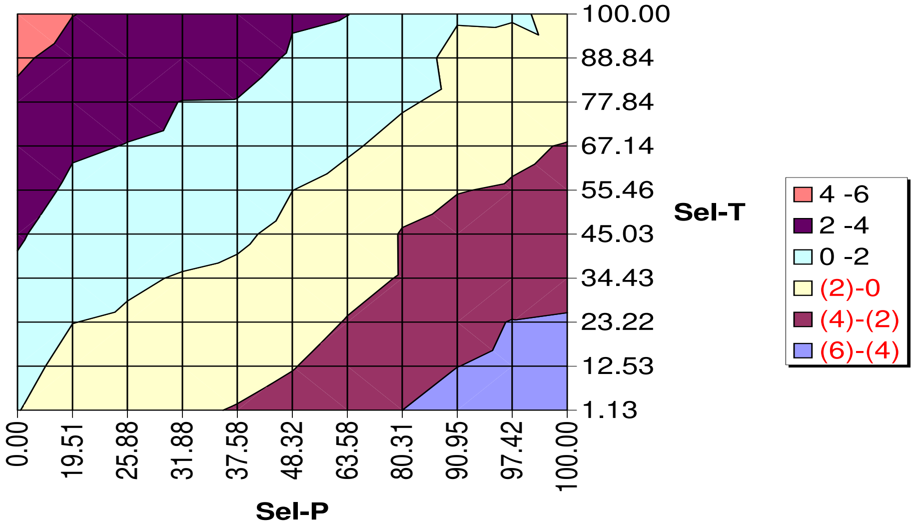


Figure 7: Time for $\sigma_P(\sigma_T(r))$ minus time for $\sigma_T(\sigma_P(r))$ without indices

The same conclusion holds when r is indexed. In this case, we created probabilistic table indexes *Table-L_r* and *Table-T_r* on r. Thus, the system used *Table-T_r* to compute $\sigma_P(\sigma_T(r))$, and used *Table-L_r* to compute $\sigma_T(\sigma_P(r))$. The results are shown in Figure 8. In conclusion, when we need to perform a series of selects on r, we should determine their execution order according to our estimates of their selectivities with respect to r, regardless of the kind of select being considered. However, if these selectivities are approximately equal, then we can give preference to data selects over temporal selects, and temporal selects over probabilistic selects.

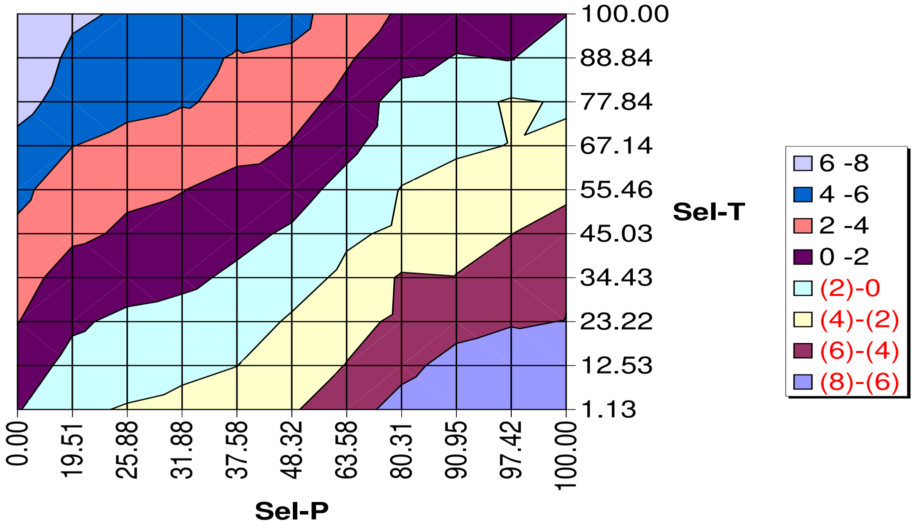


Figure 8: Time for $\sigma_P(\sigma_T(r))$ minus time for $\sigma_T(\sigma_P(r))$ with indices

Pushing Temporal Selections into Join: Note that the query $\sigma_T(r \bowtie_\alpha r')$ can be evaluated in four possible ways: A = $\sigma_T(r \bowtie_\alpha r')$, B = $\sigma_T(r) \bowtie_\alpha \sigma_T(r')$, C = $r \bowtie_\alpha \sigma_T(r')$, or D = $\sigma_T(r) \bowtie_\alpha r'$. Our goal is to determine when we should use each of these methods. To generate sample relations for r and r', we used the parameters given in Figure 11.

TP-rel	NO_TUPLES	F2_Range	C_R	AVG_TIME_PTS_R	L_R	U_R	Delta
r_1	585	1-100	2000-2090	1-10	0.1-0.5	0.1-1.0	mix
r_2	375	1-100	2000-2090	1-10	0.1-0.5	0.1-1.0	mix
r_3	748	1-1000	2000-2090	1-10	0.1-0.5	0.1-1.0	mix
r_4	346	1-1000	2000-2090	1-10	0.1-0.5	0.1-1.0	mix

Table 11: TP-Relations Used In Join-Temporal Selection Rewrite Rules

We varied T in the same way as for temporal selections and we did not vary the conjunction strategy α as this does not affect the relative running times of options A,B,C, D. We varied the number of TP-cases returned by the join by considering the queries $\sigma_T(r_1 \bowtie_\alpha r_2)$ (where $|r_1 \bowtie_\alpha r_2| = 238$), and $\sigma_T(r_3 \bowtie_\alpha r_4)$ (where $|r_3 \bowtie_\alpha r_4| = 31$). The results for these two queries are shown, respectively, in Figures 9 and 10.

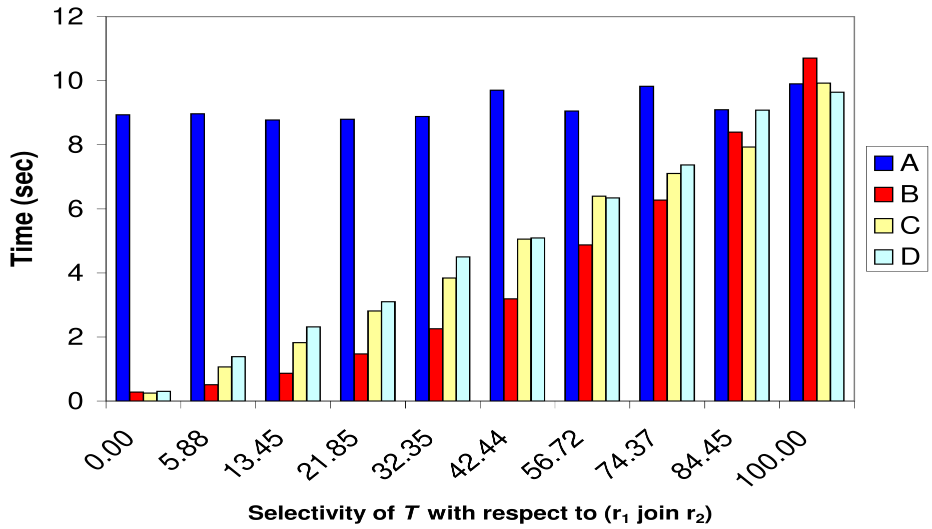


Figure 9: A = $\sigma_T(r_1 \bowtie_\alpha r_2)$ vs. B = $\sigma_T(r_1) \bowtie_\alpha \sigma_T(r_2)$ vs. C = $r_1 \bowtie_\alpha \sigma_T(r_2)$ vs. D = $\sigma_T(r_1) \bowtie_\alpha r_2$

Figure 9 demonstrates the case when $|r \bowtie_\alpha r'|$ is relatively large. Here, it is better to push selection into the join for both r and r' (i.e., strategy B) unless the selectivity of the selection predicate is very low (i.e., unless the selectivity of T is a large percentage). This result should not be surprising. As $|r \bowtie_\alpha r'|$ increases, the preference for strategy B vs. strategy A intensifies. Conversely, as $|r \bowtie_\alpha r'|$ decreases, the threshold for T where A becomes preferable to B decreases. An example is shown in Figure 10. Note that when the join removes a fairly large number of tuples, strategy C or D may perform better than strategy B (the choice of C or D depends on the selectivity of T with respect to r' and r). This occurs because strategy B requires three operations whereas the other strategies only need two.

Pushing Probabilistic Selections into Join: The query $\sigma_P(r \Join_\alpha r')$ where P is of the form $L \geq p$ for some probability $p \in [0,1]$ can be evaluated as A $= \sigma_P(r \Join_\alpha r')$ or B $= \sigma_P(\sigma_P(r) \Join_\alpha \sigma_P(r'))$. To determine when to use these options, we used TCP to generate eight TP-relations with the following parameters:

TP-rel	NO_TUPLES	C_R	AVG_TIME_PTS_R	L_R	U_R	Delta
$\{r_i \mid i \in [1,6]\}$	1000	2000-2090	1-10	0.1-0.5	0.1-1.0	mix
$\{r_i \mid i \in [7,8]\}$	10000	2000-2090	1-10	0.1-0.5	0.1-1.0	mix

The F2_Range parameter was 1-1000 for r_1 and r_2, 1-100 for r_3 and r_4, 1-10 for r_5 and r_6, and 1-10000 for r_7 and r_8. Thus, $|r_1 \Join_{pc} r_2| = 113$, $|r_3 \Join_{pc} r_4| = 1188$, $|r_5 \Join_{pc} r_6| = 11732$, and $|r_7 \Join_{pc} r_8| = 1191$. To vary P, we used probabilistic predicate $(L \geq 0.32 \cdot (\frac{1}{2})^i)$ for all $i \in [0,6]$. The results are shown in Figure 11. Note that the conjunction strategy α remained constant. This is acceptable since other experiments have verified that the running times depend on $|\sigma_P(r \Join_\alpha r')|$ instead of the choice for α. Nonetheless, note that when $\alpha \neq \alpha'$, then it is often the case that $|\sigma_P(r \Join_\alpha r')| \neq |\sigma_P(r \Join_{\alpha'} r')|$.

For each graph in Figure 11, the times for strategy A remain more or less constant. This is because the times to perform the joins are constant, and these times overwhelm the times it takes to perform the selections. Furthermore in each graph, the times for strategy B increase as the selectivity of P increases. We are interested in the threshold for P where the time for strategy B exceeds the time for strategy A. For graphs (a), (b), and (c) of Figure 11, this occurs when the selectivity of P is around 20%, 65%, and 75% respectively. Thus if $|r \Join_\alpha r'|$ increases while $\max(|r|, |r'|)$ remains constant, then the threshold for P increases. Graph (d) of Figure 11, uses base TP-relations r_7, r_8 where $|r_7| = |r_8| = 10 \cdot (|r_1| = \ldots = |r_6|)$. Here, the threshold for P is around 20%. This matches

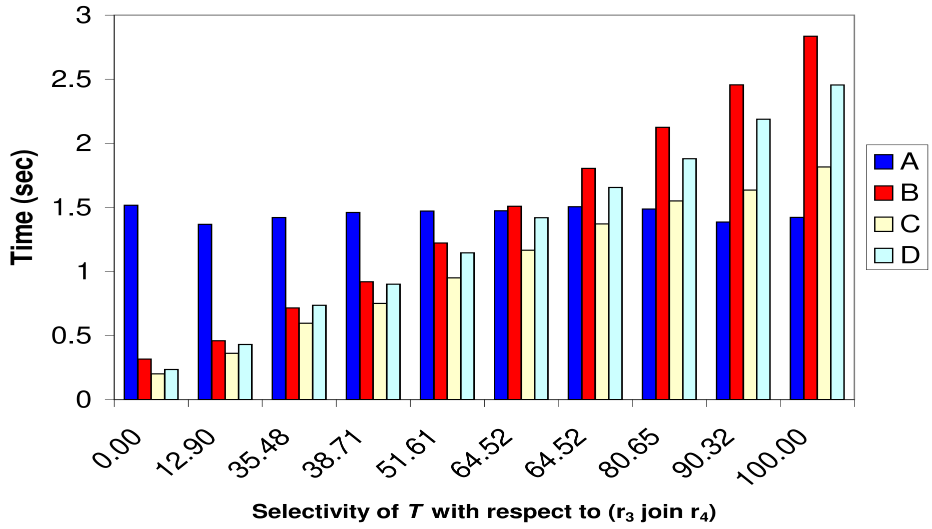
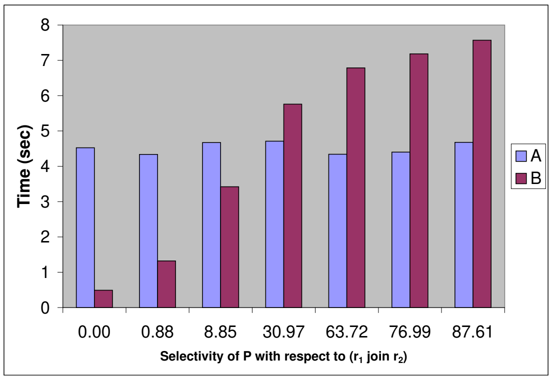
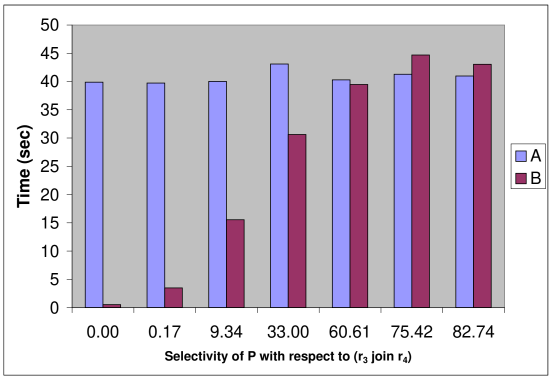


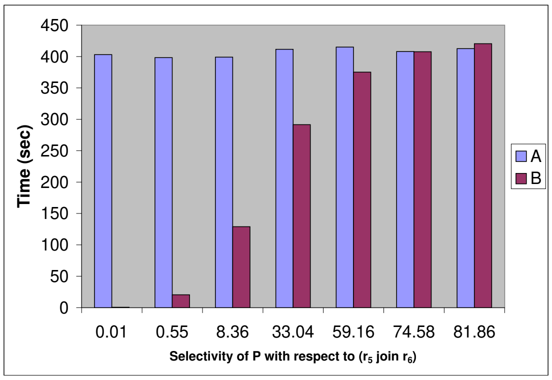
Figure 10: A $= \sigma_T(r_3 \Join_\alpha r_4)$ vs. B $= \sigma_T(r_3) \Join_\alpha \sigma_T(r_4)$ vs. C $= r_3 \Join_\alpha \sigma_T(r_4)$ vs. D $= \sigma_T(r_3) \Join_\alpha r_4$



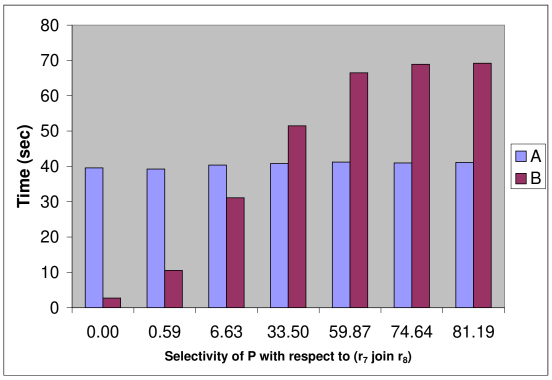
(a)



(b)



(c)



(d)

Figure 11: A = $\sigma_P(r_1 \bowtie_{pc} r_2)$ vs. B = $\sigma_P(\sigma_P(r_1) \bowtie_{pc} \sigma_P(r_2))$

the threshold for graph (a) but does not match the threshold for graph (b). Thus to estimate the threshold for P, we cannot just consider $|r \bowtie_{\alpha} r'|$ (e.g., $|r_7 \bowtie_{pc} r_8| \approx |r_3 \bowtie_{pc} r_4|$). Instead, consider the ratio between $|r \bowtie_{\alpha} r'|$ and $\max(|r|, |r'|)$ (e.g., $\frac{|r_7 \bowtie_{pc} r_8|}{\max(|r_7|,|r_8|)} \approx \frac{|r_1 \bowtie_{pc} r_2|}{\max(|r_1|,|r_2|)}$).

In conclusion, there are many cases where it is better to use strategy B even though it involves performing selects three times more often than strategy A. The accuracy of predicting when these cases occur strongly depends on the accuracy of the cardinality estimators.

6.0.5 Effectiveness of The Query Optimizer

In this section, we report on experiments conducted by us to determine the overall effectiveness of our TP query optimizer. We wanted to determine how "good" or "bad" the plans picked by the TP-query optimizer are. We used TCP to create 9 TP-relations having the properties shown in Table 12.

TP-rel	NO_TUPLES	F2_Range	C_R	AVG_TIME_PTS_R	L_R	U_R	Delta
r1	1000	100	1900-2000	0-10	0.1-0.5	0.1-1.0	mix
r2	100	100	2000-2100	2-2	0.1-0.5	0.1-1.0	mix
r3	100	100	2000-2500	2-2	0.1-0.5	0.1-1.0	mix
r4	100	100	2000-3000	2-2	0.1-0.5	0.1-1.0	mix
r5	100	100	2000-2100	5-5	0.1-0.5	0.1-1.0	mix
r6	100	100	2000-2500	5-5	0.1-0.5	0.1-1.0	mix
r7	100	100	2000-3000	5-5	0.1-0.5	0.1-1.0	mix
r8	100	100	2000-2500	10-10	0.1-0.5	0.1-1.0	mix
r9	100	100	2000-3000	10-10	0.1-0.5	0.1-1.0	mix

Table 12: TP-relations used in the second set of experiments.

Simple Queries: We first tried to see what happens with queries involving only one TP-operation. We compared compared the actual running times with the optimizer's estimated execution times. Table 13 contains the queries we tried, together with their actual and estimated execution times. It can be seen that the optimizer estimates exhibit an acceptable amount of accuracy, consistent with the accuracy of traditional cost based query optimizers. In those cases where the actual and estimate differ, the difference appears to be due to inaccurate selectivity estimates. However, the optimizer still seems to pick very good plans and seems to avoid very bad ones.

Multiple Selection Queries: We wanted to study the behavior of our optimizer when select queries involve all three types of selects (data, temporal, probabilistic). We created three queries shown in Table 14. There are 10 different feasible query plans for each of these queries. Figure 12 enumerates, these plans, shows their running times, and uses a *star* to indicate which plan was picked by the optimizer. As seen from the figure, the optimizer was able to choose the cheapest plan for queries 1 and 2, and the second best plan for query 3. The difference between the cheapest plan and the one the optimizer picked was 0.3 secs.

Multiple Joins: Next, we studied how our optimizer behaves when queries with two joins are executed. Table 15 shows four queries created by TCP. In each query, we chose TP-relations with

Query	Actual time	Estimated Time
$\sigma_{L>0.2}(r_1)$	0.47s	0.637s
$\sigma_{L>0.05}(r_1)$	1.12s	1.169s
$\sigma_{L=0.4}(r_1)$	0.156s	0.494s
$\sigma_{L<0.1}(r_1)$	1.254s	1.643s
$\sigma_{L<0.01}(r_1)$	0.635s	0.855s
$\sigma_{1900\sim1990}(r_1)$	1.304s	1.506s
$\sigma_{1950\sim2000}(r_1)$	0.979s	1.209s
$\sigma_{1970\sim1980}(r_1)$	0.405s	0.977s
$\sigma_{1960\sim2020}(r_1)$	0.776s	1.129s
$\sigma_{r_1.\mathrm{f1}=r_4.\mathrm{f1}}(r_1 \times_{ig} r_4)$	0.56s	0.765s
$\sigma_{r_2.\mathrm{f2}=r_3.\mathrm{f2}}(r_2 \times_{ig} r_3)$	0.601s	0.764s
$\sigma_{r_5.\mathrm{f2}=r_6.\mathrm{f2}}(r_5 \times_{ig} r_6)$	0.562s	0.769s
$\sigma_{r_6.\mathrm{f1}=r_8.\mathrm{f1}}(r_6 \times_{ig} r_8)$	0.593s	0.76s
$\sigma_{r_8.\mathrm{f2}=r_9.\mathrm{f2}}(r_8 \times_{ig} r_9)$	0.622s	0.763s

Table 13: Some Simple Queries, Their Estimated and Actual Running Times

#	Query
1	$\sigma_{(\mathrm{f2}<250)\wedge(1920\sim1990)\wedge(L<0.01)}(r_1)$
2	$\sigma_{(\mathrm{f2}<250)\wedge(1970\sim1990)\wedge(L<0.2)}(r_1)$
3	$\sigma_{(\mathrm{f2}<10)\wedge(1920\sim1980)\wedge(L<0.3)}(r_1)$

Table 14: Selection Queries

varying number of average time points. There are three ways of executing these two join operation. Again, we executed those plans, and examined the optimizer's choice in each case. The results are shown in Figure 13' and the optimizer's choice is shown with a *star*. The reader will not that the optimizer picked the best query plan in all four cases.

#	Query
1	$\sigma_{(r_1.\mathrm{f2}=r_6.\mathrm{f2})\wedge(r_6.\mathrm{f2}=r_9.\mathrm{f2})}(r_1 \times_{ig} r_6 \times_{ig} r_9)$
2	$\sigma_{(r_5.\mathrm{f2}=r_6.\mathrm{f2})\wedge(r_6.\mathrm{f2}=r_1.\mathrm{f2})}(r_5 \times_{ig} r_6 \times_{ig} r_1)$
3	$\sigma_{(r_1.\mathrm{f2}=r_2.\mathrm{f2})\wedge(r_2.\mathrm{f2}=r_7.\mathrm{f2})}(r_1 \times_{ig} r_2 \times_{ig} r_7)$
4	$\sigma_{(r_2.\mathrm{f2}=r_6.\mathrm{f2})\wedge(r_6.\mathrm{f2}=r_9.\mathrm{f2})}(r_2 \times_{ig} r_6 \times_{ig} r_9)$

Table 15: Join Queries

Selection/Join Mix: We created three queries involving a join, a temporal and a probabilistic select operation. Recall that temporal selects can be pushed into the join, but probabilistic selects cannot. Moreover, temporal selection can be pushed into the first argument, or the second argument or both arguments of the join. We can execute the temporal select by using an index or without

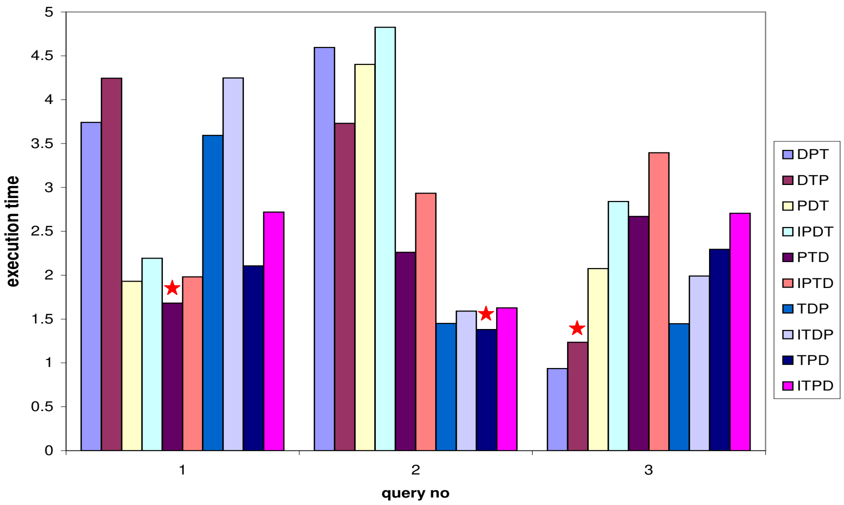


Figure 12: Ordering of Data, Temporal and Probabilistic Selects

an index. Finally, we can commute temporal and probabilistic selects. Hence, there are 8 different ways of executing these queries; in 6 of those temporal selects are executed before the join, and in 2 of them they are executed after the join. Figure 14 shows the actual execution times of these 8 plans for the three queries provided in Table 16. Once again, we marked the optimizer's choice with a *star*. Although the optimizer was not able to pick the cheapest plan for the first query, it was able to avoid very bad plans and furthermore, the difference in execution times between the plan chosen by the optimizer and the best plan was 0.311 secs. The optimizer chose the second best plan for the second query, and was able to pick the best plan for the last query.

#	**Query**
1	$\sigma_{(r_2.\mathrm{f2}=r_1.\mathrm{f2})\wedge(2000\sim 2010)\wedge(L>0.05)}(r_2 \times_{ig} r_1)$
2	$\sigma_{(r_5.\mathrm{f2}=r_8.\mathrm{f2})\wedge(2100\sim 2150)\wedge(L<0.4)}(r_5 \times_{ig} r_8)$
3	$\sigma_{(r_2.\mathrm{f2}=r_5.\mathrm{f2})\wedge(2400\sim 2500)\wedge(L<0.02)}(r_2 \times_{ig} r_5)$

Table 16: Join Queries with Selections

Finally, we made the previous join/selection mix queries more complex and created the queries shown in Table 17. For the first and the third queries, the optimizer applied 674 rewrite rules to generate a plan. The most promising plans and the one chosen by the optimizer are shown in Figure 15. The optimizer was able to choose the best plan for the first two queries, and it chose a good plan for the third query. The difference between the execution times of the best plan and the plan the optimizer chose for query 3 was 0.067secs.

We conclude this section by reporting that our cost models are reasonably accurate and easy to compute and that the optimizer does its job effectively. It quickly selects a query execution plan that is very good, while consistently avoiding bad plans.

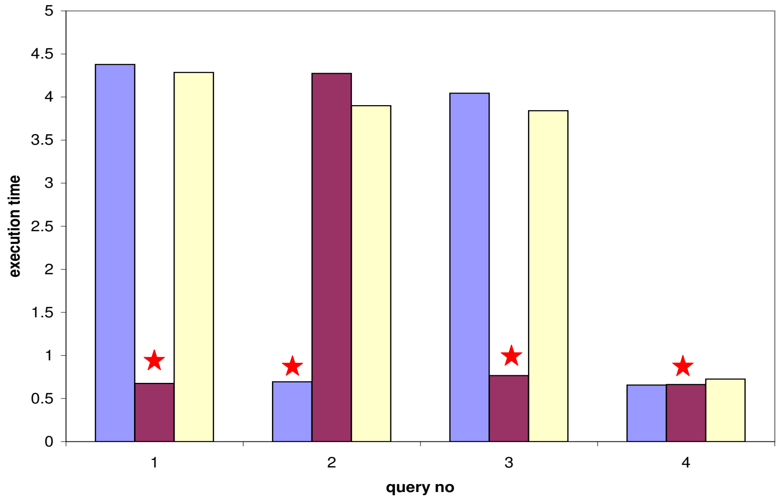


Figure 13: Ordering of Two Joins

#	**Query**
1	$\sigma_{(r_6.\mathrm{f2}=r_4.\mathrm{f2}) \wedge (r_6.\mathrm{f2}=r_8.\mathrm{f2}) \wedge (r_6.\mathrm{f2}<50) \wedge (r_4.\mathrm{f2}<75) \wedge (r_8.\mathrm{f2}>25) \wedge (2500\sim2520) \wedge (L<0.1)}(r_6 \times_{ig} r_4 \times_{ig} r_8)$
2	$\sigma_{(r_2.\mathrm{f2}=r_3.\mathrm{f2}) \wedge (r_3.\mathrm{f2}=r_5.\mathrm{f2}) \wedge (r_2.\mathrm{f2}<25) \wedge (r_3.\mathrm{f2}=5) \wedge (2000\sim2070)}(r_2 \times_{ig} r_3 \times_{ig} r_5)$
3	$\sigma_{(r_1.\mathrm{f2}=r_7.\mathrm{f2}) \wedge (r_7.\mathrm{f2}=r_8.\mathrm{f2}) \wedge (r_7.\mathrm{f2}<75) \wedge (2100\sim2400) \wedge (L<0.3)}(r_1 \times_{ig} r_7 \times_{ig} r_8)$

Table 17: More Complex Queries

7 Related Work

Dyreson and Snodgrass [7] 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.

Later, [5] extended the work of Dyreson and Snodgrass [7] in the following ways. First, they proposed TP-cases as a way of using constraints and probabilities together to store probabilistic temporal data. Then they developed an algebra extending the relational algebra to manipulate this data. They used probability intervals (hence capturing point probabilities of Dyreson and Snodgrass [7] as a special case). They allowed users to specify in their queries, arbitrary knowledge the user's may have about the dependencies between events, and in fact, their algebraic operators were parameterized by these dependency assumptions. Specifically, all independence assumptions used in

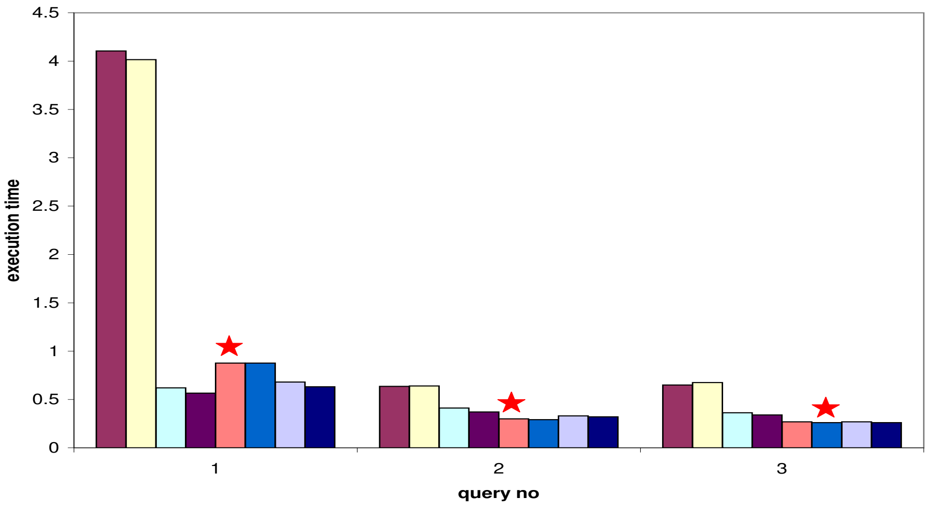


Figure 14: Ordering of a Join with Temporal and Probabilistic Selects

[7] were eliminated in [5]. They also proposed a set of operations such as compaction that were not considered elsewhere. However, they did not incorporate some things that Dyreson and Snodgrass could handle. For instance, they assumed tuples have only one indeterminate temporal attribute while [7] allows more than one. Furthermore, they do not have analogs of correlation credibility or ordering plausibility introduced by [7] and they use methods to store probability distributions provided by Dyreson and Snodgrass [7].

This paper *directly* builds on the previous two works. Specifically, it provides cost models for TP-databases which apply in large part to both the preceding works, and it provides estimates of cardinality and other statistical variables useful for query optimization. In addition, it derives a useful set of equivalence results that may be used for query rewriting. Using the software of [5], it builds what is to our knowledge, the first query optimizer for temporal probabilistic databases.

The work also extends a host of work on probabilistic (non temporal !) databases. Kiessling et. al.'s DUCK system [15, 17] provides an elegant, logical, axiomatic theory for rule based uncertainty. Building on past work of Kifer and his colleagues [16], Lakshmanan and Sadri [22] show how selected probabilistic strategies can be used to extend the previous probabilistic models. Lakshmanan and Shiri [23] have shown how deductive databases may be parameterized through the use of conjunction and disjunction strategies. Barbara et al. [1] develop a point probabilistic data model and propose probabilistic operators. When performing joins, they assume that Bayes' rule applies (and hence, as they admit up front, they make the assumption that all events are independent). Also, as they point out, unfortunately their definition leads to a "lossy" join. Cavallo and Pittarelli [3]'s important probabilistic relational database model uses probabilistic projection and join operations, but the other relational algebra operations are not specified. Also, a relation in their model is analogous to a single tuple in the framework of [1]. Dey and Sarkar [6] propose an elegant `1NF` approach to handling probabilistic databases. They support (i) having uncertainty about some objects but certain information about others, (ii) first normal form which is easy to understand and use, (iii)

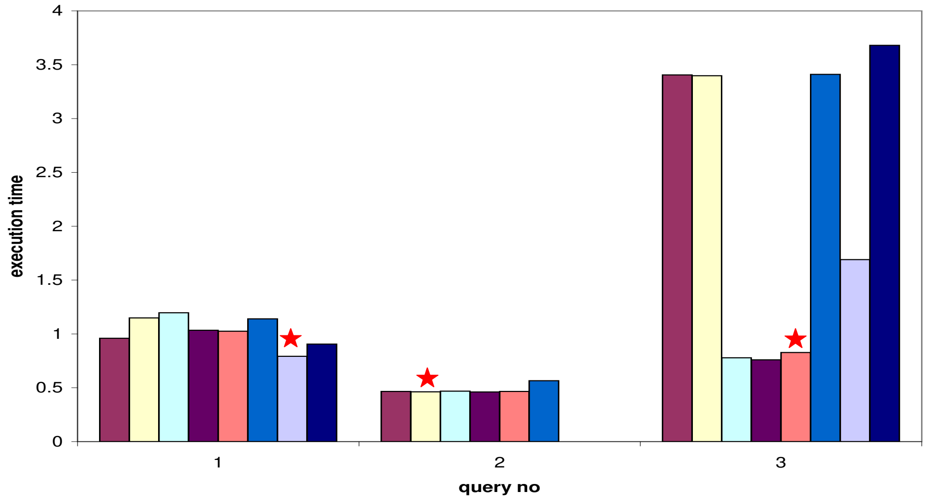


Figure 15: Plans for Complex Queries

elegant new operations like conditionalization. The `1NF` representation used by them is a special case of the annotated representation of [5] who showed experimentally that when dealing the temporal uncertainty, the TP-representation is superior. Also, the semantics of some of the operations in [6] is different from the semantics of the same operations in TPA. Another important work is the ProbView system for probabilistic databases by Lakshmanan et. al. [20]. ProbView extends the classical relational algebra by allowing users to specify in their query, what probabilistic strategy (or strategies) should be used to parameterize the query. ProbView removed the independence assumption of previous works. However, ProbView has no notion of time, 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 a temporal database uses seconds as it lowest level of temporal granularity, this gives rise to $345,600$ cases to represent just one statement — something that would quickly overwhelm ProbView. The work reported in this paper specifically builds upon the idea of a probabilistic conjunctive strategy of [20] and applies it to TP-databases. But in addition to these works, none of which provided statistical estimates of various parameters/variables associated with TP-data, we do so, and we provide a cost model for such data and a query optimizer whose sole goal is to avoid the problems faced by purely probabilistic database systems attempting to cope with huge amounts of uncertainty.

There is also much work in the temporal database community that deals with uncertainty. Snodgrass was one of the first to model indeterminate instances in his doctoral dissertation [31] — he proposed the use of a model based on three valued logic. Dutta [9] and Dubois and Prade [8] later used a fuzzy logic based approach to handle *generalized temporal events* — events that may occur multiple times. Gadia [10] proposes an elegant model to handle incomplete temporal information as well. He models values that are completely known, values that are unknown but are known to have occurred, values that are known if they have occurred, and values that are unknown even if they occurred. However, he makes no use of probabilistic information.

Koubarakis [19] proposes the use of constraints for representing event occurrences. His framework allows stating the facts that event e_1 occurred between 8 and 11 AM, and that event e_2 occurs after 12pm. From this, we may conclude that event e_2 occurs after e_1. While TP-tuples support this via queries, they do not explicitly encode this data into tuples, which Koubarakis can do.

As mentioned earlier, though many of these works touch upon uncertainty, they do not provide the comprehensive probability analysis provided in [7, 5]. They also do not propose statistical variables to maintain information about TP-relations, and corresponding cost models which form the key contributions of this paper.

The only comparable work in the area of query processing and query optimization comes from the field of temporal databases. Salzberg and Tsotras provide an excellent overview of temporal indexing methods developed in the community over the past decade [27]. Query optimization and cost models for temporal databases have been studied in early 90s by Gunadhi and Segev [13, 14], and their study was later continued by Soo, Snodgrass, Jensen and Slivinskas [33, 30]. Their research was primarily concentrated on designing efficient algorithms for relational operations on temporal databases which minimized I/O costs. These results, however, are not directly applicable to the TP-Databases, due to the complex structure of the data model developed in [5]. Moreover, to our knowledge, this is the first attempt to develop a detailed model for estimating the cost and cardinality of probabilistic selections.

8 Conclusions

Databases that contain nondeterministic temporal information are growing in importance. Important work in this field was started by Dyreson and Snodgrass [7] who proposed a probabilistic temporal model of such data. Dekhtyar et. al [5] extended the framework of [7] by proposing an extension of the relational algebra for such data, as well as by eliminating many assumptions about the data that was present in prior work.

In this paper, we make additional contributions to the area of temporal probabilistic (TP) databases. First, we develop a TP calculus that is equivalent to the TP algebra in expressive power. Second, we develop a large set of equivalence results in such databases that constitute rewrite rules that a query optimizer might use. Third, we develop a cost model for TP databases. Fourth, we have implemented our cost model and our rewrite rules in a prototype optimizer for TP databases. Using this implementation, we have evaluated (i) the accuracy of our cost model, (ii) the effectiveness of our rewrite rules, and (iii) the effectiveness of our TP optimizer as a whole.

References

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

[2] M.H. Böhlen, R.T. Snodgrass, and M.D. Soo. (1996) Coalescing in Temporal Databases, in *Proc. VLDB'96*, pp. 180-191.

[3] R. Cavallo and M. Pittarelli. (1987) The Theory of Probabilistic Databases, in *Proc. VLDB'87.*

[4] Codd, E. F. "Relational Completeness of Data Base Sublanguages", in [26], pages 65-98.

[5] A. Dekhtyar, R. Ross and V.S. Subrahmanian, Probabilistic Temporal Databases, Part I: Algebra, *ACM Transactions on Database Systems,* vol 26, 1, pps 41–95, March 2001.

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

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

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

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

[10] S. Gadia, S. Nair and Y.C. Poon. (1992) Incomplete Information in Relational Temporal Databases, in *Proc. VLDB'92.*

[11] H. Garcia-Molina, J. D. Ullman and J. Widom, *Database System Implementation*, Prentice Hall, 2000

[12] G. Graefe, (1995) The Cascades Framework for Query Optimization, in *the Bulletin of the TC on Data Engineering*, 18(3), pp 19-29.

[13] H. Gunadhi, A. Segev (1990) A Framework for Query Optimization in Temporal Databases in *Proc. Conf. on Statistical and Scientific Database Management'1990*,pp. 131-147.

[14] H. Gunadhi, A. Segev (1991) Query Processing Algorithms for Temporal Intersection Joins, in *Proc. ICDE'91*, pp. 336-344.

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

[16] M. Kifer and A. Li. (1988) *On the Semantics of Rule-Based Expert Systems with Uncertainty*, 2-nd Intl. Conf. on Database Theory, Springer Verlag LNCS 326, (eds. M. Gyssens, J. Paredaens, D. Van Gucht), Bruges, Belgium, pp. 102–117.

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

[18] G. Kollios, V.J. Tsotras: Hashing Methods for Temporal Data. To appear at IEEE Transactions on Knowledge and Data Engineering. 2001.

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

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

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

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

[23] V.S. Lakshmanan and N. Shiri. (1996) Parametric Approach with Deductive Databases with Uncertainty, in *Proc. Workshop on Logic In Databases 1996*, pp 61-81.

[24] R. Ramakrishnan, J. Gehrke. (2000) *Database Management Systems*, 2nd Ed., McGraw-Hill.

[25] S. Ross. (1998) *A First Course in Probability*, Prentice Hall, 1998.

[26] Rustin, R. [1972]. *Data Base Systems*, Prentice-Hall, New Jersey, 1972.

[27] B. Salzberg and V. Tsotras. (1999) Comparison of Access Methods for Time-Evolving Data, *ACM Computing Surveys*, vol 31, No. 2., pp. 158-221.

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

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

[30] G. Slivinskas, C.S. Jensen, R.T. Snodgrass. (2000) Query Plans for Conventional and Temporal Queries Involving Duplicates and Ordering, in *Proc. ICDE-2000*.

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

[32] R.T. Snodgrass, M. Soo, (1995) Supporting Multiple Calendars in R.T. Snodgrass (Ed.), *The TSQL2 Temporal Query Language*, pp. 103-121, Kluwer, 1995.

[33] M. D. Soo, R. T. Snodgrass, C. S. Jensen (1994) Efficient Evaluation of the Valid-Time Natural Join, in *Proc. ICDE'1994* pp. 282-29

[34] D. Zhang, V.J. Tsotras, B. Seeger: Efficient Temporal Join Processing using Indices. Proc. ICDE'02 (to appear), San Jose, CA, Feb.-March 2002.

[35] Q. Zhu and P. Larson. (1998) Solving Local Cost Estimation Problem for Global Query Optimization in Multidatabase Systems, *Journal of Distributed and Parallel Databases*, 6(4), pp. 373–421.

A Proofs for the TP-Calculus Theorems

Proof of Theorem 1: Suppose db is a TP-database, q is a TPA-query over db, and q_i is a subquery of q. We proceed by induction on the number of operators in q. This number is zero in the base case where $q = r$. Otherwise, q must obey the structure specified in Definition 19.

Suppose Q is a TPC-query over db and $Q_i = \{s_i \mid F'_i\}$ is a subquery of Q. By the inductive hypothesis, assume that TPA-query q_i can be expressed as TPC-query Q_i for all subqueries of q. Then TPA-query q can be expressed as TPC-query Q in the following way:

1. If $q = r$, then $Q = \{s \mid s \,\underline{\varepsilon}\, r\}$.
2. If $q = \sigma_{\mathcal{C}}(q_1)$ where $\mathcal{C} = \mathcal{C}_1 \wedge \ldots \wedge \mathcal{C}_n$, then $Q = \{s \mid \exists\, s_1\, (F'_1 \,\wedge\, E_1 \,\wedge\, \ldots \,\wedge\, E_n)\}$ where
   - $E_i = s.A \;\Theta\; c$ when $\mathcal{C}_i = (A \;\Theta\; c)$.
   - $E_i = s.C : s_1.C \;@\; T$ when $\mathcal{C}_i = T$.
   - $E_i = s.C : s_1.P \;\Theta\; p$ when $\mathcal{C}_i = (P \;\Theta\; p)$.
3. If $q = \pi_{\mathcal{F}}(q_1)$ where $\mathcal{F} = a_1, \ldots, a_n$, then
   $Q = \{s \mid \exists\, s_1\, (F'_1 \,\wedge\, s.a_1 = s_1.a_1 \,\wedge\, \ldots \,\wedge\, s.a_n = s_1.a_n)\}$.
4. If $q = \rho_{\mathcal{R}}(q_1)$ where $R_1 = (a_1, \ldots, a_n)$, then
   $Q = \{s \mid \exists\, s_1\, (F'_1 \,\wedge\, s.\mathcal{R}(a_1) = s_1.a_1 \,\wedge\, \ldots \,\wedge\, s.\mathcal{R}(a_n) = s_1.a_n)\}$.
5. If $q = \Xi(q_1)$, then $Q = Q_1$.
6. If $q = (q_1 \times_\alpha q_2)$ where $R_1 = (a_1, \ldots, a_n)$ and $R_2 = (a'_1, \ldots, a'_{n'})$, then
   $$Q = \{s \mid \exists\, s_1\, (F'_1 \wedge_{\{\langle\otimes,\alpha\rangle,\langle\Xi,\Xi_{id}\rangle\}} \exists\, s_2\, (F'_2 \,\wedge\, s.a_1 = s_1.a_1 \,\wedge\, \ldots \,\wedge\, s.a_n = s_1.a_n \,\wedge\, s.a'_1 = s_1.a'_1 \,\wedge\, \ldots \,\wedge\, s.a'_n = s_1.a'_{n'}))\}.$$
7. If $q = (q_1 \bowtie_\alpha q_2)$ where $R_1 = (a_1, \ldots, a_n)$ and $R_2 = (a'_1, \ldots, a'_{n'})$, then
   $$Q = \{s \mid \exists\, s_1\, (F'_1 \wedge_{\{\langle\otimes,\alpha\rangle,\langle\Xi,\Xi_{id}\rangle\}} \exists\, s_2\, (F'_2 \,\wedge\, s.a_1 = s_1.a_1 \,\wedge\, \ldots \,\wedge\, s.a_n = s_1.a_n \,\wedge\, s.a'_1 = s_1.a'_1 \,\wedge\, \ldots \,\wedge\, s.a'_n = s_1.a'_{n'}))\}.$$
8. If $q = \kappa_\chi(q_1 \cap \ldots \cap q_n)$, then $Q = \{s \mid \exists\, s_1\, (F'_1) \wedge_{\{\langle\kappa,\chi\rangle,\langle\Xi,\Xi_{id}\rangle\}} \cdots \wedge_{\{\langle\kappa,\chi\rangle,\langle\Xi,\Xi_{id}\rangle\}} \exists\, s_n\, (F'_n)\}$.
9. If $q = \kappa_\chi(q_1 \cup \ldots \cup q_n)$, then $Q = \{s \mid \exists\, s_1\, (F'_1) \vee_{\{\langle\kappa,\chi\rangle,\langle\Xi,\Xi_{id}\rangle\}} \cdots \vee_{\{\langle\kappa,\chi\rangle,\langle\Xi,\Xi_{id}\rangle\}} \exists\, s_n\, (F'_n)\}$.
10. If $q = q_1 - q_2$, then $Q = \{s \mid \exists\, s_1\, (F'_1) \wedge \neg\, \exists\, s_2\, (F'_2)\}$. $\diamond$

Proof of Theorem 2: Suppose db be a TP-database, s is a TP-variable over R, τ, $Q = \{s \mid F'\}$ is a TPC-query over db, and $Q_i = \{s_i \mid F'_i\}$ is a subquery of Q where s_i is a TP-variable over R_i, τ. We proceed by induction on the structures given in Definitions 21, 22, 23, 25, and 26.

Suppose q is a TPA-query over db and q_i is a subquery of q. By the inductive hypothesis, assume that TPC-query Q_i can be expressed as TPA-query q_i for all subqueries of Q. Then TPC-query Q can be expressed as TPA-query q in the following way:

1. If $Q = \{s \mid s \,\underline{\varepsilon}\, r\}$, then $q = r$.
2. If $Q = \{s \mid \exists\, s_1\, (F'_1 \,\wedge\, E)\}$, then $q = \sigma_{\mathcal{C}}(\pi_{\mathcal{F}}(\rho_{\mathcal{R}}(q_1)))$ where
   - $\mathcal{R}(A_1) = A$ for each TP-atom in E of the form $s.A = s_1.A_1$.
   - $\mathcal{F} = a_1, \ldots, a_k$ when $R = (a_1, \ldots, a_k)$.
   - $\mathcal{C} = \mathcal{C}_1 \wedge \ldots \wedge \mathcal{C}_n$ where

- $\mathcal{C}_i = (A \; \Theta \; c)$ when the i-th TP-atom in E is of the form $s.A \; \Theta \; c$.
- $\mathcal{C}_i = T$ when the i-th TP-atom in E is of the form $s.C : s_1.C \; @ \; T$.
- $\mathcal{C}_i = (P \; \Theta \; p)$ when the i-th TP-atom in E is of the form $s.C : s_1.P \; \Theta \; p$.
- $\mathcal{C}_i = \mathit{true}$ when the i-th TP-atom in E is of the form $s.A = s_1.A_1$.

3. If $Q = \{s \mid \exists \, s_1 \, (F_1' \wedge_\Omega \exists \, s_2 \, (F_2' \wedge \mathcal{L}))\}$ where $\Omega = \{\langle \otimes, \alpha \rangle, \langle \Xi, \beta \rangle\}$, then $q = \pi_{\mathcal{F}}(\rho_{\mathcal{R}_1}(q_1) \bowtie_\alpha \rho_{\mathcal{R}_2}(q_2))$ where
   - $\mathcal{R}_1(A_1) = A$ for each TP-linker in $\mathcal{L}$ of the form $s.A = s_1.A_1$. Otherwise, $\mathcal{R}_1(A_1) = A_1$.
   - $\mathcal{R}_2(A_2) = A$ for each TP-linker in $\mathcal{L}$ of the form $s.A = s_2.A_2$. Otherwise, $\mathcal{R}_2(A_2) = A_2$.
   - $\mathcal{F} = a_1, \ldots, a_k$ when $R = (a_1, \ldots, a_k)$.
4. If $Q = \{s \mid \exists \, s_1 \, (F_1') \; \theta \; \ldots \; \theta \; \exists \, s_n \, (F_n')\}$ where $\Omega = \{\langle \kappa, \chi \rangle, \langle \Xi, \beta \rangle\}$, then
   - $q = \kappa_\chi(q_1 \cap \ldots \cap q_n)$ when $\theta = \wedge_\Omega$.
   - $q = \kappa_\chi(q_1 \cup \ldots \cup q_n)$ when $\theta = \vee_\Omega$.
   - $q = q_1 - \ldots - q_n$ when $\theta = \wedge\neg$. $\diamond$

B Proofs of Equivalence Results

The proofs of query equivalence results contained in this section are all done using *Theoretical Annotated Temporal Algebra (TATA)*, defined in [5] on *annotated relations*, the flat relational equivalents of TP-relations. Each annotated relation consists of annotated tuples which have the form (d, t, L_t, U_t) where d is the data part of the tuple (a collection of relational attributes similar to the data part of a TP-tuple), t is a single time point, and $[L_t, U_t] \subseteq [0, 1]$ is a probability interval. The annotation operation can be applied to TP-tuples to produce "equivalent" annotated relations.

TATA has been defined in [5] on annotated relations. Except for TP-compression, which is specific to the format of TP-tuples, all other TPA operations have their analogs in TATA. It has been shown in [5] there that TPA operations *correctly implement* the semantics of corresponding TATA operations, i.e. the equivalences $\mathrm{ANN}(Op(r)) \equiv Op(\mathrm{ANN}(r))$ and $\mathrm{ANN}(Op(r, r')) \equiv Op(\mathrm{ANN}(r), \mathrm{ANN}(r'))$ hold for unary and binary operations respectively of the TPA and TATA.

As annotated tuples are flat, reasoning in terms of annotations of TP-relations in proofs is less cumbersome. The *correct implementation* theorems of [5], assure us that a query equivalence holds in TPA iff it holds in TATA. Hence, we choose to prove equivalences in TATA to make our reasoning more transparent and make the proofs shorter.

Proof of Theorem 3.

1. Selection. We prove the theorem statement for atomic selection conditions. Theorem 4 will ensure that it holds for conjunctive selection conditions. Part 5 of this theorem will ensure that it holds for disjunctive selection conditions.

   Three cases need to be considered.

   - Selection condition is on *data*. Then the idempotence of selection in TPA follows from idempotence of selection in classical relational algebra [11].

- Selection condition C is *temporal*. Let $at = (d, t, L, U) \in ANN(\sigma_C(r))$. This means that timepoint t satisfies selection condition C. But then, $ANN(\sigma_C(\sigma_C(r)))$ will also contain at, hence $\sigma_C(r) \subseteq \sigma_C(\sigma_C(r))$. The other subset inclusion, $\sigma_C(\sigma_C(r)) \subseteq \sigma_C(r)$ follows from the definition of selection in TPA.
- Selection condition C is *probabilistic*. Let $at = (d, t, L, U) \in ANN(\sigma_C(r))$. This means that the probability interval $[L, U]$ satisfies selection condition C. But then, $ANN(\sigma_C(\sigma_C(r)))$ will also contain at, hence $\sigma_C(r) \subseteq \sigma_C(\sigma_C(r))$. The other subset inclusion, $\sigma_C(\sigma_C(r)) \subseteq \sigma_C(r)$ follows from the definition of selection in TPA.

2. Projection. Direct corollary of idempotence of projection in classical relational algebra [11]: projection in TPA affects only the data part of the tp-tuples.

3. Compaction. In [5] compaction operation is defined to have the following three properties:

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

   Idempotence of compaction, therefore, follows from Compactness and NFA properties.

4. Intersection. By definition $(r \cap_{\kappa_\chi} r) = \kappa_\chi(r \cap r)$. Let $at = (d, t, L, U) \in ANN(r)$. Then $ANN(r \cap r) \equiv ANN(r) \cap ANN(r)$ will contain *two distinct instances* of at. As r is compact, $ANN(r)$ contains no other tuples for the pair d, t, and therefore, $ANN(r \cap r')$ will only contain two distinct instances of at for the pair d, t. By Identity property of combination functions, $\chi(\{[L, U], [L, U]\}) = [L, U]$, therefore, $ANN(\kappa_\chi(r \cap r)) \equiv \kappa_\chi(ANN(r \cap r))$ will contain the tuple $at' = (d, t, L, U)$. As $at' = at$, we have shown that $r \subseteq r \cap_{\kappa_\chi} r$.

   Consider now an annotated tuple $at = (d, t, L, U) \in ANN(r \cap_{\kappa_\chi} r)$. By definition of intersection, r has to contain some tuple $at' = (d, t, L', U')$, and as we know that r is compact, this tuple will be unique for the pair d, t. But then, by the Identity property of the combination function, $ANN(r \cap_{\kappa_\chi} r)$ will contain the tuple $at'' = (d, t, L', U')$. As $r \cap_{\kappa_\chi} r$ is compact, $at'' = at$, i.e., $L' = L$ and $U' = U$. But then $at = at'$ and therefore $at \in ANN(r)$. Hence, $r \cap_{\kappa_\chi} r \subseteq r$.

5. Union. By definition $(r \cup_{\kappa_\chi} r) = \kappa_\chi(r \cup r)$. Let $at = (d, t, L, U) \in ANN(r)$. Then $ANN(r \cup r) \equiv ANN(r) \cap ANN(r)$ will contain *two distinct instances* of at. As r is compact, $ANN(r)$ contains no other tuples for the pair d, t, and therefore, $ANN(r \cap r')$ will only contain two distinct instances of at for the pair d, t. By Identity property of combination functions, $\chi(\{[L, U], [L, U]\}) = [L, U]$, therefore, $ANN(\kappa_\chi(r \cup r)) \equiv \kappa_\chi(ANN(r \cup r))$ will contain the tuple $at' = (d, t, L, U)$. As $at' = at$, we have shown that $r \subseteq r \cap_{\kappa_\chi} r$.

   Consider now an annotated tuple $at = (d, t, L, U) \in ANN(r \cup_{\kappa_\chi} r)$. By definition of union, r has to contain some tuple $at' = (d, t, L', U')$, and as we know that r is compact, this tuple will be unique for the pair d, t. But then, by the Identity property of the combination function, $ANN(r \cup_{\kappa_\chi} r)$ will contain the tuple $at'' = (d, t, L', U')$. As $r \cup_{\kappa_\chi} r$ is compact, $at'' = at$, i.e., $L' = L$ and $U' = U$. But then $at = at'$ and therefore $at \in ANN(r)$. Hence, $r \cup_{\kappa_\chi} r \subseteq r$.

6. Difference. Consider an annotated tuple $at = (d, t, L, U) \in ANN(r)$. We show $at \in ANN(r - r')$ iff $at \in ANN((r - r') - r')$. Let r'' denote $r - r'$. Two cases are possible:

   - There exists an annotated tuple $at' = (d, t, L', U') \in ANN(r')$. In this case, by definition of difference, $at \notin ANN(r'')$. Also, as $ANN(r'' - r') \subseteq r$ (see definition of difference), $at \notin ANN(r'' - r')$.
   - There is no tuple in $ANN(r)$ for the pair d, t. In this case $at \in ANN(r'') = ANN(r - r')$. But then, at also has to be in $ANN(r'' - r)$. $\diamond$

Proof of Theorem 4.

1. Selection. See Theorem 13 in [5].

2. Projection. We prove $\pi_{\mathcal{F}}(\pi_{\mathcal{G}}(r)) \equiv \pi_{\mathcal{G}\cap\mathcal{F}}$. The other equivalence will follow.

   Let $tp = (d, \gamma) \in r$. Let $d' = \pi_{\mathcal{F}}(d)$. By definition of classical projection, d' will only contain attributes in $\mathcal{F}$. Let $d'' = \pi_{\mathcal{G}}(d')$. Then, d'' will contain only attributes from $\mathcal{G}$. Hence, d'' will contain only attributes from $\mathcal{F} \cap \mathcal{G}$, i.e., $d'' = \pi_{\mathcal{F}\cap\mathcal{G}}(d)$.

   By definition of projection in TPA, $\pi_{\mathcal{F}}(r)$ will contain tp-tuple $tp' = (\pi_{\mathcal{F}}(d), \gamma) = (d', \gamma)$ and $\pi_{\mathcal{G}}(\pi_{\mathcal{F}}(r))$ will contain tp-tuple $tp'' = (\pi_{\mathcal{G}}(d'), \gamma) = (d'', \gamma)$. But also, $\pi_{\mathcal{F}\cap\mathcal{G}}(r)$ will contain tp-tuple $tp^{\star} = (\pi_{\mathcal{F}\cap\mathcal{G}}(d), \gamma) = (d'', \gamma) = tp''$. So, for every tuple $tp = (d, \gamma) \in r$ both $\pi_{\mathcal{G}}(\pi_{\mathcal{F}}(r))$ and $\pi_{\mathcal{F}\cap\mathcal{G}}(r)$ will contain the same tuple $tp'' = (d'', \gamma)$. Therefore, $\pi_{\mathcal{G}}(\pi_{\mathcal{F}}(r)) \equiv \pi_{\mathcal{F}\cap\mathcal{G}}(r)$.

3. Intersection. $r \cap_{\kappa} r' = \kappa(r \cap r')$. Thus in order to show the commutativity of intersection, we need to show commutativity of *multiset intersection*: $r \cap r' \equiv r' \cap r$.

   By definition of multiset intersection in TATA

$$ANN(r \cap r') = ANN(r) \cap ANN(r') =$$
$$\{at = (t, d, L, U) \in ANN(r) | (\exists at' \in ANN(r'))(at' = (t, d, L', U')\} \uplus$$
$$\{at' = (t, d, L', U') \in ANN(r') | (\exists at \in ANN(r))(at = (t, d, L, U)\}$$

   and

$$ANN(r' \cap r) = ANN(r') \cap ANN(r) =$$
$$\{at' = (t, d, L', U') \in ANN(r) | (\exists at \in ANN(r))(at = (t, d, L, U)\}$$
$$\uplus\{at = (t, d, L, U) \in ANN(r) | (\exists at' \in ANN(r))(at' = (t, d, L', U')\}.$$

   As $\uplus$, the multiset union operation is commutative, the above two expressions are equivalent and hence, $ANN(r \cap r') \equiv ANN(r' \cap r)$, yielding immediately $r \cap r' \equiv r' \cap r$.

4. Union. $r \cup_{\kappa} r' = \kappa(r \cup r')$. Thus in order to show the commutativity of intersection, we need to show commutativity of *multiset union*: $r \cup r' \equiv r' \cup r$.

   By definition of multiset union in TATA

$$ANN(r \cup r') = ANN(r) \cup ANN(r') = ANN(r) \uplus ANN(r')$$

   and

$$ANN(r' \cap r) = ANN(r') \cap ANN(r) = ANN(r') \uplus ANN(r).$$

As $\uplus$, the multiset union operation (of set theory) is commutative, the above two expressions are equivalent and hence, $ANN(r \cup r') \equiv ANN(r' \cup r)$, yielding immediately $r \cup r' \equiv r' \cup r$.

5. Difference

   We show that $(r - r') - r'' \equiv r - (r' \cup r'')$, the other equivalences follow from it.

   • $\underline{(r - r') - r'' \subseteq r - (r' \cup r'')}$.

   Let $at = (d, t, L, U) \in ANN((r - r') - r'')$. Then by definition of difference, $at \in ANN(r - r')$ and $ANN(r'')$ contains *no* tuple for the pair d, t. As $at \in ANN(r - r')$, it must be the case that $at \in ANN(r)$ and $ANN(r')$ contains *no* tuple for the pair d, t.

   But then $ANN(r' \cup r'')$ will contain *no* tuples containing data part d and timepoint t. As $at \in ANN(r)$, we get $at \in ANN(r) - ANN(r' \cup r'') = ANN(r - (r' \cup r))$.

   • $\underline{r - (r' \cup r'') \subseteq (r - r') - r''}$.

   Let $at = (d, t, L, U) \in ANN(r - (r' \cup r'')$. Then, by definition of difference, $at \in r$ and there is no tuple containing the pair d, t in $ANN(r' \cup r'')$. Hence, there is no annotated tuple containing the pair d, t in *eiter* $ANN(r')$ or $ANN(r'')$. Then $at \in ANN(r) - ANN(r') = ANN(r - r')$ and consequently, $at \in ANN(r - r') - ANN(r'') = ANN((r - r') - r'')$.

6. Cartesian Product.

   Recall that given two data tuples d and d' we consider tuples (d, d') and (d', d) to be *equivalent*.

   Let $at = (d, t, L, U) \in r$ and $at' = (d', t, L', U') \in r'$. Then, $\text{ANN}(r \times_\alpha r')$ will contain the tuple $at^\star = (d, d', t, L^\star, U^\star)$ where $[L^\star, U^\star] = [L, U] \otimes_\alpha [L', U']$. Also $\text{ANN}(r' \times_\alpha r)$ will contain the tuple $at^* = (d', d, t, L^*, U^*)$, where $[L^*, U^*] = [L', U'] \otimes_\alpha [L, U]$. By commutativity of $\otimes$ operation, $[L, U] \otimes_\alpha [L', U'] = [L', U'] \otimes [L, U]$, i.e. $[L^\star, U^\star] = [L^*, U^*]$. But then $at^\star = at^*$. Therefore, for each pair of tuples in $\text{ANN}(r)$ and $\text{ANN}(r')$ *the same* tuple will be found in $\text{ANN}(r \times_\alpha r'$ and $\text{ANN}(r' \times_\alpha)$ (if the tuples have the same timepoint). This proves the theorem.

7. Join.

   Remember that $r \bowtie_\alpha r' = \pi_{\mathcal{F}}(\sigma_{\mathcal{C}}(r \times_\alpha r'))^2$. But by commutativity of cartesian product, $r \times_\alpha r' \equiv r' \times_\alpha r$, and therefore

$$r \bowtie_\alpha r' = \pi_{\mathcal{F}}(\sigma_{\mathcal{C}}(r \times_\alpha r')) \equiv \pi_{\mathcal{F}}(\sigma_{\mathcal{C}}(r' \times r)) = r' \bowtie_\alpha r.$$

$\diamond$

Proof of Theorem 5.

1. Cartesian Product.

   • $(r \times_\alpha r') \times_\alpha r'' \subseteq r \times_\alpha (r' \times_\alpha r'')$.

   Let $at^\star = (d^\star, t, L^\star, U^\star) \in \text{ANN}((r \times_\alpha r') \times_\alpha r)$. Then, by definition of cartesian product, there exist annotated tuples $at_1 = (d_1, t, L_1, U_1) \in \text{ANN}(r \times_\alpha r')$ and $at'' = (d'', t, L'', U'') \in \text{ANN}(r'')$ such that $d^* = (d_1, d'')$ and $[L^*, U^*] = [L_1, U_1] \otimes_\alpha [L'', U'']$. Also, as $at_1 \in \text{ANN}(r \times_\alpha r')$, there exist tuples $at = (d, t, L, U) \in \text{ANN}(r)$ and $at' = (d', t, L', U'] \in \text{ANN}(r')$ such that $d_1 = (d, d')$ and $[L_1, U_1] = [L, U] \otimes_\alpha [L', U']$.

[2] Ignoring the renaming of the attributes.

But then, $\mathrm{ANN}(r' \times_\alpha r'')$ will contain a tuple $at_2 = (d_2, t, L_2, U_2)$ where $d_2 = (d', d'')$ and $[L_2, U_2] = [L', U'] \otimes_\alpha [L'', U'']$ and $\mathrm{ANN}(r \times_\alpha (r' \times_\alpha r''))$ will contain a tuple $at^* = (d^*, t, L^*, U^*)$ where $d^* = (d, d_2) = (d, d', d'') = d^\star$ and $[L^*, U^*] = [L, U] \otimes_\alpha [L_2, U_2] = [L, U] \otimes ([L', U'] \otimes_\alpha [L'', U'']) = ([L, U] \otimes_\alpha [L', U']) \otimes_\alpha [L'', U''] = [L^\star, U^\star]$ by associativity of $\otimes$. But then, since $[L^*, U^*] = [L^\star, U^\star]$ we get $at^\star = a^*$ which proves the inclusion.

- $r \times_\alpha (r' \times_\alpha r'') \subseteq (r \times_\alpha r') \times_\alpha r''$.

Let $at^\star = (d^\star, t, L^\star, U^\star) \in \mathrm{ANN}(r \times_\alpha (r' \times_\alpha r))$. Then, by definition of cartesian product, there exist annotated tuples $at_2 = (d_2, t, L_2, U_2) \in \mathrm{ANN}(r' \times_\alpha r'')$ and $at = (d, t, L, U) \in \mathrm{ANN}(r)$ such that $d^\star = (d, d_2)$ and $[L^\star, U^\star] = [L, U] \otimes_\alpha [L_2, U_2]$. Also, as $at_2 \in \mathrm{ANN}(r' \times_\alpha r'')$, there exist tuples $at' = (d', t, L', U') \in \mathrm{ANN}(r')$ and $at'' = (d'', t, L'', U''] \in \mathrm{ANN}(r'')$ such that $d_2 = (d', d'')$ and $[L_2, U_2] = [L', U'] \otimes_\alpha [L'', U'']$.

But then, $\mathrm{ANN}(r \times_\alpha r')$ will contain a tuple $at_1 = (d_1, t, L_1, U_1)$ where $d_1 = (d, d')$ and $[L_1, U_1] = [L, U] \otimes_\alpha [L', U']$ and $\mathrm{ANN}((r \times_\alpha r') \times_\alpha r'')$ will contain a tuple $at^* = (d^*, t, L^*, U^*)$ where $d^* = (d_1, d'') = (d, d', d'') = d^\star$ and $[L^*, U^*] = [L_1, U_1] \otimes_\alpha [L'', U''] = ([L, U] \otimes [L', U']) \otimes_\alpha [L'', U''] = [L, U] \otimes_\alpha ([L', U']) \otimes_\alpha [L'', U'']) = [L^\star, U^\star]$ by associativity of $\otimes$. But then, since $[L^*, U^*] = [L^\star, U^\star]$ we get $at^\star = a^*$ which proves the inclusion and the theorem.

2. Join.

Let R, R' and R'' be the lists of data attributes for tp-relations r, r' and r'' respectively.

By definition of join

$$(r \bowtie_\alpha r') \bowtie_\alpha r'' = \pi_{R \cup (R'-R)}(\sigma_{\wedge_{a \in R \cap R'} a = \mathcal{R}(a)}(r \times_\alpha r') \bowtie_\alpha r'' =$$

$$\pi_{(R \cup (R'-R)) \cup (R'' - (R \cup (R'-R)))}(\sigma_{\wedge_{a \in (R \cup (R'-R)) \cap R''} a = \mathcal{R}(a)}(\pi_{R \cup (R'-R)}(\sigma_{\wedge_{a \in R \cap R'} a = \mathcal{R}(a)}(r \times_\alpha r')) \times_\alpha r'')) =$$

(by pushing selection through projection (Theorem 6, proved below) and pushing selection through cartesian product (Theorem 8, proved below))

$$\pi_{(R \cup (R'-R)) \cup (R'' - (R \cup (R'-R)))}(\pi_{R \cup (R'-R)}(\sigma_{\wedge_{a \in (R \cup (R'-R)) \cap R''} a = \mathcal{R}(a)}(\sigma_{\wedge_{a \in R \cap R'} a = \mathcal{R}(a)}((r \times_\alpha r') \times_\alpha r'')))) =$$

$$\pi_{(R \cup (R'-R) \cup (R'' - (R \cup (R'-R)))) \cap (R \cup (R'-R))}(\sigma_{\wedge_{a \in ((R \cup (R'-R)) \cap R'')) \cup (R \cap R')} a = \mathcal{R}(a)}((r \times_\alpha r') \times_\alpha r'')).$$

Similar reasoning leads to

$$r \bowtie_\alpha (r' \bowtie_\alpha r'') = \pi_{(R \cup ((R' \cup (R''-R)) - R)) \cap (R \cup R' \cup (R''-R'))}(\sigma_{\wedge_{a \in (R \cap (R' \cup (R''-R'))) \cup (R' \cap R'')} a = \mathcal{R}(a)}((r \times_\alpha (r' \times_\alpha r'')).$$

By associativity of cartesian product $(r \times_\alpha r') \times_\alpha r'' \equiv r \times_\alpha (r' \times_\alpha r'')$.

We will now show that the selection conditions and projection lists in the two expressions above are equivalent.

Projection lists:

$\underline{(R \cup (R' - R) \cup (R'' - (R \cup (R' - R)))) \cap (R \cup (R' - R))} = ((R \cup (R' - R)) \cap R) \cup ((R'' - (R \cup (R' - R))) \cap R'') \cup ((R \cup (R' - R)) \cap (R' - R)) \cup ((R'' - (R \cup (R' - R))) \cap (R' - R)) \cup ((R \cup (R' - R)) \cap R'') \cup ((R'' - (R \cup (R' - R))) \cap R'') = R \cup ((R'' - R) - ((R \cap R) \cup ((R' - R) \cap R))) \cup (R' - R) \cup ((R'' \cap (R' - R)) - (R' - R)) \cup ((R \cap R') \cup (R'' \cap (R' - R))) \cup ((R'' \cap R'') - ((R" \cap R) \cup (R'' \cap (R' - R)))) =$

$$R \cup \emptyset \cup (R' - R) \cup \emptyset \cup (R \cap R'') \cup (R'' \cap (R' - R)) \cup (R'' - ((R'' \cap R) \cup (R'' \cap (R' - R)))) = R \cup (R' - R) \cup ((R'' - R) \cap (R'' - R')) = \underline{R \cup (R' - R) \cup ((R'' - R') - R))}.$$

$$\underline{(R \cup ((R' \cup (R'' - R)) - R)) \cap (R \cup R' \cup (R'' - R'))} = (R \cap R) \cup (R \cap R') \cup (R \cap (R'' - R')) \cup (((R' \cup (R'' - R')) - R) \cup R) \cup (((R' \cup (R'' - R')) - R) \cup R') \cup (((R' \cup (R'' - R')) - R) \cap (R'' - R)) = R \cup (R \cap R') \cup (R \cap (R'' - R')) \cup (((R' \cup (R'' - R')) \cap R) - (R \cap R)) \cup (((R' \cap R') \cup ((R'' - R') \cap R')) - (R \cap R')) \cup (((R' \cap (R'' - R')) \cup ((R'' - R') \cap (R'' \cap R')) - (R \cap (R'' - R'))) = R \cup (R \cap R') \cup (R \cap (R'' - R')) \cup ((R' \cap R) - R) \cap (R' - (R \cap R')) \cap (((R \cap (R'' - R')) \cup (R'' - R')) - (R \cap (R'' - R'))) = R \cup (R \cap R') \cup (R \cap (R'' - R')) \cup (R' - R) \cup ((R'' - R') - (R \cap (R'' - R'))) = \underline{R \cup (R' - R) \cup ((R'' - R') - R)}.$$

Selection conditions:

$$\underline{((R \cup (R' - R)) \cap R'') \cup (R \cap R')} = (R \cup (R' - R) \cup (R \cap R')) \cap (R'' \cup (R \cap R')) = (R \cap R'') \cup ((R' - R) \cap R'') \cup (R \cap R' \cap R'') \cup (R \cap R') \cup ((R' - R) \cap R \cap R') \cup ((R \cap R') \cap (R \cap R')) = (R \cap R'') \cup (R'' \cap (R' - R)) \cup (R \cap R') \cup (R \cap R') \cup (R \cap R' \cap R'') = (R \cap R') \cup (R \cap R'') \cup (R'' \cap (R - R')) = (R \cap R') \cup ((R \cap R'') - R') \cup (R \cap R' \cap R'') \cup ((R'' \cap R') - R) = \underline{(R \cap R') \cup (R' \cap R'') \cup (R \cap (R'' - R'))}.$$

$$\underline{(R \cap (R' \cup (R'' - R')) \cup (R' \cap R''))} = (R \cup (R' \cap R'')) \cap (R' \cup (R'' - R') \cup (R' \cap R'')) = (R \cap R') \cup (R \cap (R'' - R)) \cup (R \cap R' \cap R'') \cup ((R' \cap R'') \cap (R' \cap R'')) \cup ((R' \cap R'') \cap R') \cup ((R' \cap R'') \cap (R'' - R')) = \underline{(R \cap R') \cup (R' \cap R'') \cup (R \cap (R'' - R'))}.$$

Combined, the commutativity of the cartesian product and the equivalence of the projection lists and selection conditions prove the associativity of join. ◇

Proof of Proposition 1.

1. $r \cap_\kappa r' \subseteq r \cup_\kappa r'$.

   Let $at = (d, t, L, U) \in \mathrm{ANN}(r \cap_\kappa r')$. We need to show $at \in r \cup_\kappa r'$. By definition of intersection, r (and hence, $\mathrm{ANN}(r)$) is compact. Therefore, at is the only tuple in $\mathrm{ANN}(r \cap_\kappa r')$ for the pair d, t. By definitions of intersection and compaction, we infer that there must be annotated tuples $at_i = (d, t, L_i, U_i) \in \mathrm{ANN}(r \cap r')$, $1 \leq i \leq n$, $n \geq 2$, such that $\kappa(\{[L_i, U_i] | 1 \leq i \leq n\}) = [L, U]$. But then, by definition of multiset intersection, each at_i must belong to either $\mathrm{ANN}(r)$ or $\mathrm{ANN}(r')$ (with at least one tuple in each relation). Without loss of generality, assume that $at_1, \ldots, at_r \in \mathrm{ANN}(r)$ and $at_{r+1}, \ldots at_n \in \mathrm{ANN}(r')$, for some $1 \leq r \leq n - 1$. But then, by definition of multiset union, all at_i belong to $\mathrm{ANN}(r \cup r')$ and *no* other annotated tuple for the pair d, t belongs to $\mathrm{ANN}(r \cup r')$. Therefore, by definitions of the union and compaction $\mathrm{ANN}(r \cup_\kappa r')$ will contain tuple $at' = (d, t, L', U')$ where $[L', U'] = \kappa(\{[L_i, U_i] | 1 \leq i \leq n\})$. But then, $[L', U'] = [L, U]$ and therefore $at' = at$, i.e., $at \in \mathrm{ANN}(r \cup_\kappa r')$.

2. $r \cap r' \not\equiv r - (r - r')$.

   In short: $r \cap r'$ will contain tuples from both r and r', while $r - (r - r')$ by definition of difference will contain only tuples from r.

   To be more specific, consider two annotated tuples $at = (d, t, L, U) \in \mathrm{ANN}(r)$ and $at' = (d, t, L', U') \in \mathrm{ANN}(r')$. By definition of $r \cap r'$, $\mathrm{ANN}(r \cap r')$ will contain both at and at'.

   On the other hand, $\mathrm{ANN}(r - r')$ will contain *no* tuple for the pair (d, t), as $at \in \mathrm{ANN}(r)$ and $at' \in \mathrm{ANN}(r')$. Therefore, $\mathrm{ANN}(r - (r - r'))$ will contain at, as $at \in \mathrm{ANN}(r)$ and no tuple for d, t is in $\mathrm{ANN}(r - r')$. However, as $at' \notin \mathrm{ANN}(r)$, $at' \notin \mathrm{ANN}(r - (r - r'))$.

3. $r - (r' \cap r'') \equiv (r - r') \cup (r - r'')$.

• $\underline{r - (r' \cap r'') \subseteq (r - r') \cup (r - r'')}$.
Let $at = (d, t, L, U) \in \mathrm{ANN}(r - (r' \cap r''))$. Then $at \in \mathrm{ANN}(r)$ and there is no tuple in $\mathrm{ANN}(r \cap r'')$ which would contain the d, t pair. Thus, two possibilities have to be considered: (i) neither $\mathrm{ANN}(r')$, nor $\mathrm{ANN}(r'')$ contain tuples for the d, t pair and (ii) one of the two relations (either $\mathrm{ANN}(r')$ or $\mathrm{ANN}(r'')$) contains a tuple $at' = (d, t, L', U')$.

In case (i) both $\mathrm{ANN}(r - r')$ and $\mathrm{ANN}(r - r'')$ will contain at, and therefore $\mathrm{ANN}((r - r') \cup (r - r''))$ will contain at.

In case (ii) (assume, for simplicity that $\mathrm{ANN}(r')$ contains tuple at' as above; the other case is symmetric), $\mathrm{ANN}(r - r'')$ will contain at while $\mathrm{ANN}(r - r')$ won't. But then, $at \in \mathrm{ANN}((r - r') \cup (r - r''))$.

• $\underline{r - (r' \cap r'') \supseteq (r - r') \cup (r - r'')}$.
Let, now $at = (d, t, L, U) \in \mathrm{ANN}((r - r') \cup (r - r''))$. Then either $at \in \mathrm{ANN}(r - r')$ or $at \in \mathrm{ANN}(r - r'')$ or at is in both $\mathrm{ANN}(r - r')$ and $\mathrm{ANN}(r - r'')$.

Let $at \in \mathrm{ANN}(r - r')$ and $at \notin \mathrm{ANN}(r - r'')$ (the case when $at \in \mathrm{ANN}(r - r'')$ and $at \notin \mathrm{ANN}(r - r')$ is symmetric). Then, $at \in \mathrm{ANN}(r)$ and $\mathrm{ANN}(r')$ contains no tuple $at' = (d, t, L', U')$. Also, as $at \in \mathrm{ANN}(r)$, it has to be the case that $\mathrm{ANN}(r'')$ contains some tuple $at'' = (d, t, L'', U'')$. However, as no tuple for d, t is in $\mathrm{ANN}(r')$, $\mathrm{ANN}(r \cap r')$ will also contain no tuple for d, t and therefore $\mathrm{ANN}(r - (r' \cap r'')$ will contain at.

Now, in the case when both $\mathrm{ANN}(r - r')$ and $\mathrm{ANN}(r - r'')$ contain at, we know that $at \in \mathrm{ANN}(r)$ and neither $\mathrm{ANN}(r)$ nor $\mathrm{ANN}(r'')$ contains any tuples for the pair d, t. Then, $\mathrm{ANN}(r' \cap r'')$ will contain no tuple for d, t either, and therefore $at \in \mathrm{ANN}(r - (r' \cap r''))$.

4. $r - (r' \cup r'') \equiv (r - r') \cap (r - r'')$.

• $\underline{r - (r' \cup r'') \subseteq (r - r') \cap (r - r'')}$.
Let $at = (d, t, L, U) \in \mathrm{ANN}(r - (r' \cup r''))$. Then $at \in \mathrm{ANN}(r)$ and there is no tuple in $\mathrm{ANN}(r \cup r'')$ which would contain the d, t pair. Therefore, neither $\mathrm{ANN}(r')$ nor $\mathrm{ANN}(r'')$ contain any tuple for d, t. But then, both $\mathrm{ANN}(r - r')$ and $\mathrm{ANN}(r - r'')$ will contain at and therefore, so will $\mathrm{ANN}((r - r') \cap (r - r''))$.

• $\underline{r - (r' \cup r'') \supseteq (r - r') \cap (r - r'')}$.
Let, now $at = (d, t, L, U) \in \mathrm{ANN}((r - r') \cap (r - r''))$. Then $at \in \mathrm{ANN}(r - r')$ and $at \in \mathrm{ANN}(r - r'')$, and therefore $ar \in \mathrm{ANN}(r)$ and no tuple in either $\mathrm{ANN}(r')$ or $\mathrm{ANN}(r'')$ will contain the pair d, t. But then $\mathrm{ANN}(r' \cup r'')$ will contain no such tuple as well, and because $at \in \mathrm{ANN}(r)$, $at \in \mathrm{ANN}(r - (r' \cup r''))$. $\diamond$

Proof of Theorem 6.

We prove this theorem for atomic constraints C. Then by Theorem 13 from [5] this will also hold for non-atomic constraints. For an atomic constraint C three cases are possible.

- C is a <u>data</u> constraint. In this case the statement of the theorem is true since a similar statement is true in classical relational algebra. The TP-case field of r is not "touched" by either $\mathcal{F}$ of C, and hence both selection and projection on r will behave exactly as they would in the absence of the TP-case.

- C is a temporal constraint. We know that $\mathcal{F}$ contains only data fields in it. We also know that C does not refer to any data field. It is easy to see from this that the statement of the theorem holds.

  Indeed, let $at = (d, t, l, u)$ be an annotated tuple from $ANN(\sigma_C(\pi_{\mathcal{F}}(r)))$. Since C is a temporal constraint, we know that t satisfies C. Since $ANN(\sigma_C(r)) \subseteq ANN(r)$, we know that $at \in ANN(\pi_{\mathcal{F}}(r))$. Since $ANN(\pi_{\mathcal{F}}(r)) = \pi_{\mathcal{F}}(ANN(r))$, there exists an annotated tuple $at' = (d', t, l, u) \in ANN(r)$ such that $d = \mathcal{P}(d')$. But then, since t satisfies C, $at' \in \sigma_C(ANN(r))$ and therefore, $at \in \pi_{\mathcal{F}}(\sigma_C(ANN(r)))$.

  Going the other way, let $at = (d, t, l, u) \in \pi_{\mathcal{F}}(\sigma_C(ANN(r)))$. Then, $\sigma_C(ANN(r))$ contains an annotated tuple $at' = (d', t, l, u)$ where $d = \mathcal{P}(d')$. Since $(\sigma_C(ANN(r))) \subseteq (ANN(r))$, $at' \in ANN(r)$, and therefore $at \in \pi_{\mathcal{F}}(ANN(r))$. But $at' \in \sigma_C(ANN(r))$ implies that t satisfies C, and therefore it must be the case that $at \in \sigma_C(\pi_{\mathcal{F}}(ANN(r)))$.

- C is a probabilistic constraint. As in the case above, $\mathcal{F}$ contains only data fields while C refers only to the contents of the TP-case field. Applying reasoning similar to the case of temporal constraints we can establish that the statement of the theorem is true in this case as well. $\diamond$

Proof of Theorem 7.

- $\sigma_C(\kappa_\chi(r) \subseteq \kappa_\chi(\sigma_C(r))$.

  Let $at = (d, t, l, u) \in ANN(\sigma_C(\kappa_\chi(r))$. We know that at satisfies C, and as C is either a data or temporal constraint, we know that any annotated tuples $at'' = (d, t, l'', u'')$ will also satisfy C.

  As at satisfies C and since $ANN(\sigma_C(r')) = \sigma_C(ANN(r'))$ for any tp-relation r', we know that $at \in ANN(\kappa_\chi(r))$. By the property of Conservativeness of κ_χ there exists at least one annotated tuple $at' = (d, t, l', u') \in ANN(r)$. Now, let $ANN(r)[d, t] = \{at_1, \ldots, at_n\}$, $(\forall 1 \leq i \leq n)(at_i = (d, t, l_i, u_i))$. We know that $[l, u] = \chi(\{[l_1, u_1], \ldots, [l_n, u_n]\})$. As it was noticed above all $at' \in ANN(r)[d, t]$ satisfy C, and therefore, $ANN(r)[d, t] \subseteq ANN(\sigma_C(r))$. But as $ANN(\sigma_C(r)) \subseteq ANN(r)$, $ANN(r)[d, t] = ANN(\sigma_C(r))[d, t]$. Therefore, $at = (d, t, l, u)$ will be in $ANN(\kappa_\chi(\sigma_C(r)))$ which means that $\sigma_C(\kappa_\chi(r) \subseteq \kappa_\chi(\sigma_C(r))$.

- $\sigma_C(\kappa_\chi(r) \supseteq \kappa_\chi(\sigma_C(r))$.

  Let $at = (d, t, l, u) \in ANN(\kappa_\chi(\sigma_C(r)))$. By the Conservativeness property of κ_χ operation we know that there exists at least one annotated tuple $tp' = (d, t, l', u') \in ANN(\sigma_C(r))$. Let now $ANN(\sigma_C(r))[d, t] = \{at_1, \ldots, at_n\}$, where for all $1 \leq i \leq n$ $at_i = (d, t, l_i, u_i)$ and $[l, u] = \chi(\{[l_1, u_1], \ldots, [l_n, u_n]\})$. We know that all these annotated tuples satisfy C. Also, since for any tp-relation r and selection condition C' $ANN(\sigma_{C'}(r)) \subseteq ANN(r)$, $ANN(\sigma_C(r))[d, t] \subseteq ANN(r)$. But since C is a data or temporal constraint, we know that all annotated tuples in the set $ANN(r)[d, t]$ must satisfy it. Therefore $ANN(r)[d, t] = ANN(\sigma_C(r))[d, t]$.

  ¿From the latter equality and the fact that $[l, u] = \chi(\{[l_1, u_1], \ldots, [l_n, u_n]\})$ we get that $at = (d, t, l, u) \in ANN(\kappa_\chi(r))$. But we also know that at satisfies C as would any annotated tuple with data part d and temporal part t. Therefore $at \in ANN(\sigma_C(\kappa_\chi(r)))$ which means that $\sigma_C(\kappa_\chi(r) \supseteq \kappa_\chi(\sigma_C(r))$.

As you may notice, the key step in the proof was the fact that in a tp-relation r, for any data part d and timepoint t it was true that $ANN(\sigma_C(r))[d,t] = ANN(r)[d,t]$ for temporal or data select conditions C. This may not be true in the case when select condition is probabilistic. Therefore the statement of the theorem above is not true for probabilistic select conditions. $\diamond$

Proof of Theorem 8.

1. $\underline{\sigma_{C_1}(r \times_\alpha r') \equiv \sigma_{C_1}(r) \times_\alpha r'}$. We break the proof of this fact into two parts.

   - $\sigma_{C_1}(ANN(r \times_\alpha r')) \subseteq \sigma_{C_1}(ANN((r)) \times_\alpha ANN(r')$.
     Let $at'' = (d'', t, l, u) \in \sigma_{C_1}(ANN(r \times_\alpha r'))$. Clearly, $at'' \in ANN(r \times_\alpha r')$. As we know ([5], Theorem 16), $ANN(r \times_\alpha r') = ANN(r) \times_\alpha ANN(r')$, hence $at'' = (d'', t, l, u) \in ANN(r) \times_\alpha ANN(r')$. Then, by the definition of cartesian product on annotated relations, there exist two annotated tuples $at = (d, t, l_1, u_1) \in ANN(r)$ and $at' = (d', t, l_2, u_2) \in ANN(r')$ such that $d = d, d'$ and $[l, u] = [l_1, u_1] \otimes_\alpha [l_2, u_2]$. As $at'' \in \sigma_{C_1}(ANN(r \times_\alpha r'))$ we know that d'' satisfies C_1. But since the only fields mentioned in C_1 are those from the relational schema of r (and $ANN(r)$), it must be the case that d satisfies C_1 as d is the part of d''. But then, $at \in \sigma_{C_1}(r)$, and therefore $at'' \in \sigma_{C_1}(ANN(r)) \times_\alpha ANN(r')$.
   - $\sigma_{C_1}(ANN(r \times_\alpha r')) \supseteq \sigma_{C_1}(ANN((r)) \times_\alpha ANN(r')$.
     Going the other way, we assume that $at'' = (d'', t, l, u) \in \sigma_{C_1}(ANN(r)) \times_\alpha ANN(r')$. Clearly, then there exist two tuples $at = (d, t, l_1, u_1) \in ANN(r)$ and $at' = (d', t, l_2, u_2)\sigma_{C_1}(ANN(r'))$ such that $d = d, d'$ and $[l, u] = [l_1, u_1] \otimes_\alpha [l_2, u_2]$. As $at \in \sigma_{C_1}(ANN(r'))$ we also know that $at \in ANN(r)$. But then $at'' \in ANN(r) \times_\alpha ANN(r') = ANN(r \times_\alpha r')$. Also, since $at \in \sigma_{C_1}(ANN(r'))$ we know that d satisfies C_1. Therefore, since C_1 contains only references to the fields from the relational schema of r $(ANN(r))$, d'' also satisfies C_1 and therefore $at'' \in \sigma_{C_1}(ANN(r \times_\alpha r'))$.

2. $\underline{\sigma_{C_2}(r \times_\alpha r') \equiv r \times_\alpha \sigma_{C_2}(r')}$.
   The proof of this part of the theorem is symmetric to the proof of part 1.

3. $\underline{\sigma_C(r \times_\alpha r') \equiv \sigma_C(r) \times \sigma_C(r')}$.
   We prove this statement similarly to the proof of the statement in part 1. $\diamond$

Proof of Theorem 9.

Let $\mathcal{F}$ be the list of join attributes of r and r' and C' be the join condition. Then,

1. $\sigma_{C_1}(r \bowtie_\alpha r') = \sigma_{C_1}(\pi_\mathcal{F}(\sigma_{C'}(r \times_\alpha r'))) \equiv \pi_\mathcal{F}(\sigma_{C_1}(\sigma_{C'}(r \times_\alpha r'))) \equiv \pi_\mathcal{F}(\sigma_{C'}(\sigma_{C_1}(r \times_\alpha r'))) \equiv \pi_\mathcal{F}(\sigma_{C'}(\sigma_{C_1}(r) \times_\alpha r')) \equiv \sigma_{C_1}(r) \bowtie_\alpha r'$.

2. $\sigma_{C_2}(r \bowtie_\alpha r') = \sigma_{C_2}(\pi_\mathcal{F}(\sigma_{C'}(r \times_\alpha r'))) \equiv \pi_\mathcal{F}(\sigma_{C_2}(\sigma_{C'}(r \times_\alpha r'))) \equiv \pi_\mathcal{F}(\sigma_{C'}(\sigma_{C_2}(r \times_\alpha r'))) \equiv \pi_\mathcal{F}(\sigma_{C'}(r \times_\alpha \sigma_{C_2}(r'))) \equiv r \bowtie_\alpha \sigma_{C_2}(r')$.

3. $\sigma_C(r \bowtie_\alpha r') = \sigma_C(\pi_{\mathcal{F}}(\sigma_{C'}(r \times_\alpha r'))) \equiv \pi_{\mathcal{F}}(\sigma_C(\sigma_{C'}(r \times_\alpha r'))) \equiv \pi_{\mathcal{F}}(\sigma_{C'}(\sigma_C(r \times_\alpha r'))) \equiv \pi_{\mathcal{F}}(\sigma_{C'}(\sigma_C(r) \times_\alpha \sigma_C(r'))) \equiv \sigma_C(r) \bowtie_\alpha \sigma_C(r')$.

$\diamond$

Proof of Theorem 10.

1. $\sigma_{\mathcal{C}}(r \cap_\kappa r') \equiv \sigma_\kappa \mathcal{C}(r) \cap_\kappa \sigma_{\mathcal{C}}(r') \equiv r \cap_\kappa \sigma_{\mathcal{C}}(r') \equiv \sigma_{\mathcal{C}}(r) \cap_\kappa r'$.

   First we prove the statement of the theorem for multiset intersection.

   - $\underline{\sigma_C(r \cap r') \subseteq \sigma_C(r) \cap \sigma_C(r')}$.

     Let $at = (d, t, l, u) \in ANN(\sigma_C(r \cap r'))$. Then at satisfies C and as C is a data or temporal condition, so would any annotated tuple at'' with data part d and temporal part t. As $ANN(\sigma_C(r \cap r')) \subseteq ANN(r \cap r')$, $at \in r \cap r'$. Two cases are possible: (i) $at \in ANN(r)$ and (ii) $at \in ANN(r')$. We will consider case (i), the remaining case is symmetric.

     As $at \in r$ we know that there has to be at least one tuple $at' = (d, t, l'.u') \in ANN(r')$. As we have noticed above, at' satisfies C. In this case $at \in ANN(\sigma_C(r))$ and $at' \in ANN(\sigma_C(r))$. But then, $at \in ANN(\sigma_C(r)) \cap ANN(\sigma_C(r'))$, which shows that $\sigma_C(r \cap r') \subseteq \sigma_C(r) \cap \sigma_C(r')$.

   - $\underline{\sigma_C(r \cap r') \supseteq \sigma_C(r) \cap \sigma_C(r')}$. Let $at = (d, t, l, u) \in ANN(\sigma_C(r) \cap \sigma_C(r')) = ANN(\sigma_C(r)) \cap ANN(\sigma_C(r'))$. There are two possibilities: (i) $at \in ANN(\sigma_C(r))$ and (ii) $at \in ANN(\sigma_C(r'))$. We will consider the first one, proof for the second case is symmetric.

     As $at \in ANN(\sigma_C(r))$ we know that at satisfies C and also that $at \in ANN(r)$. Also, as $at \in ANN(\sigma_C(r)) \cap ANN(\sigma_C(r'))$, there has to be at least one tuple $at' = (d, t, l', u') \in ANN(\sigma_C(r'))$. Clearly, at' also satisfies C and $at' \in ANN(r')$. But in this case $\{at, at'\} \subseteq ANN(r \cap r')$ and as at satisfies C, $at \in ANN(\sigma_C(r \cap r'))$ proving that $\sigma_C(r \cap r') \supseteq \sigma_C(r) \cap \sigma_C(r')$. $\diamond$

   Now $r \cap_\kappa r' = \kappa(r \cap r')$ and therefore, (using the result of the Theorem 7), we get: $\sigma_{\mathcal{C}}(r \cap_\kappa r') = \sigma_{\mathcal{C}}(\kappa(r \cap r')) \equiv \kappa(\sigma_{\mathcal{C}}(r \cap r')) \equiv \kappa(\sigma_{\mathcal{C}}(r) \cap \sigma_{\mathcal{C}}(r')) = \sigma_{\mathcal{C}}(r) \cap_\kappa \sigma_{\mathcal{C}}(r')$. $\diamond$

2. $\sigma_{\mathcal{C}}(r \cup_\kappa r') \equiv \sigma_{\mathcal{C}}(r) \cup_\kappa \sigma_{\mathcal{C}}(r')$.

   First we prove the statement for multiset union.

   - $\underline{\sigma_C(r \cup r') \subseteq \sigma_C(r) \cup \sigma_C(r')}$.

     Let $at = (d, t, l, u) \in ANN(\sigma_C(r \cup r'))$. In this case at satisfies C and $at \in ANN(r \cup r')$. Two possibilities exist: (i) $at \in ANN(r)$ and (ii) $at \in ANN(r')$. We will consider the first one here, the proof for the other one is symmetric.

     As $at \in ANN(r)$ and at satisfies C, $at \in ANN(\sigma_C(r))$, and therefore $at \in ANN(\sigma_C(r)) \cup ANN(\sigma_C(r')) = ANN(\sigma_C(r) \cup ANN(\sigma_C(r'))$. This proves the desired subset inclusion.

   - $\underline{\sigma_C(r \cup r') \supseteq \sigma_C(r) \cup \sigma_C(r')}$. Let $at = (d, t, l, u) \in ANN(\sigma_C(r) \cup \sigma_C(r')) = ANN(\sigma_C(r)) \cup ANN(\sigma_C(r'))$. As in the previous case, two possible situations exist: (i) $at \in ANN(\sigma_C(r))$

and (ii) $at \in ANN(\sigma_C(r'))$. We consider the first possibility, the proof for the second one will be symmetric.

As $at \in ANN(\sigma_C(r))$, at satisfies C and $at \in ANN(r)$. Then $at \in ANN(r) \cup ANN(r') = ANN(r \cup r')$. As at satisfies C, we concluded that $at \in ANN(\sigma_C(r \cup r'))$. $\diamond$

Now, $r \cup_\kappa r' = \kappa(r \cup r')$ and therefore, (using the result of the Theorem 7), we get: $\sigma_{\mathcal{C}}(r \cup_\kappa r') = \sigma_{\mathcal{C}}(\kappa(r \cup r')) \equiv \kappa(\sigma_{\mathcal{C}}(r \cup r')) \equiv \kappa(\sigma_{\mathcal{C}}(r) \cap \sigma_{\mathcal{C}}(r')) = \sigma_{\mathcal{C}}(r) \cup_\kappa \sigma_{\mathcal{C}}(r')$. $\diamond$

3. $\sigma_{\mathcal{C}}(r - r') \equiv \sigma_{\mathcal{C}}(r) - \sigma_{\mathcal{C}}(r') \equiv \sigma_{\mathcal{C}}(r) - r'$.

   We prove the first equality. Second equality can be proven similarly.

   - $\underline{\sigma_C(r - r') \subseteq \sigma_C(r) - \sigma_C(r')}$. Let $at = (d, t, l, u) \in ANN(\sigma_C(r - r'))$. Then at satisfies C and $at \in ANN(r - r')$. In this case $at \in r$ and ***there are no tuples of the form*** (d, t, l', u') in r'.

     As $at \in r$ and at satisfies C, $at \in ANN(\sigma_C(r))$. Also, we know that no tuple of the form (d, t, l', u') is in $ANN(\sigma_C(r'))$. But then, $at \in ANN(\sigma_C(r)) - ANN(\sigma_C(r')) = ANN(\sigma_C(r) - \sigma_C(r')$, which proves the desired inclusion.

   - $\underline{\sigma_C(r - r') \supseteq \sigma_C(r) - \sigma_C(r')}$. Let $at = (d, t, l, u) \in ANN(\sigma_C(r) - \sigma_C(r'))$. In this case, $at \in ANN(\sigma_C(r))$ and no tuple of the form (d, t, l', u') is in $ANN(\sigma_C(r'))$. Clearly, at satisfies C and as C is either a data or a temporal constraint, any tuple of the form (d, t, l', u') would satisfy C. As $\sigma_C(r) \subset (r)$, $at \in r$. Also, we know that r' does not contain any tuples of the form (d, t, l', u') since otherwise, $\sigma_C(r')$ would have contained them. But then, $at \in ANN(r - r')$ and therefore $at \in ANN(\sigma_C(r - r'))$, which shows that $\sigma_C(r - r') \supseteq \sigma_C(r) - \sigma_C(r')$. $\diamond$

Proof of Theorem 11.

- $\sigma_C(r \times_\alpha r') \equiv \sigma_C(\sigma_C(r) \times_\alpha \sigma_C(r'))$.

  As $\mathrm{ANN}(\sigma_C(r)) \subseteq \mathrm{ANN}(r)$ and $\mathrm{ANN}(\sigma_C(r')) \subseteq \mathrm{ANN}(r')$, the $\sigma_C(r \times_\alpha r') \supseteq \sigma_C(\sigma_C(r) \times_\alpha \sigma_C(r'))$ inclusion is immediate.

  We now concentrate on proving $\sigma_C(r \times_\alpha r') \subseteq \sigma_C(\sigma_C(r) \times_\alpha \sigma_C(r'))$.

  Let $C = L \geq x$. The proof for $L > x$, $U \geq x$, $U > x$ is similar. Let $at = (d, t, L^*, U^*) \in \mathrm{ANN}(\sigma_C(r \times_\alpha r'))$. Then $L^* \geq x$, and $at \in \mathrm{ANN}(r \times_\alpha r')$. By definition of cartesian product, there exist annotated tuples $at' = (d, t, L', U') \in \mathrm{ANN}(r)$ and $at'' = (d, t, L'', U'') \in \mathrm{ANN}(r')$ such that $[L^*, U^*] = [L', U'] \otimes_\alpha [L'', U'']$.

  But, by the Bottomline axiom for probabilistic conjunction strategies, $L^* \leq \min(L', L'')$ and hence, $L' \geq x$ and $L'' \geq x$. Therefore, both at' and at'' satisfy C and $at' \in \mathrm{ANN}(\sigma_C(r))$, and $at'' \in \mathrm{ANN}(\sigma_C(r'))$. But then, $at \in \mathrm{ANN}(\sigma_C(r) \times_\alpha sigma_C(r'))$, and as at satisfies C, $at \in \mathrm{ANN}(\sigma_C(\sigma_C(r) \times_\alpha \sigma_C(r')))$. $\diamond$

- $\sigma_C(r \bowtie_\alpha r') \equiv \sigma_C(\sigma_C(r) \bowtie_\alpha \sigma_C(r'))$.

$\sigma_C(r \bowtie_\alpha r') \equiv \sigma_C(\pi_{\mathcal{F}}(\sigma_{C'}(r \times_\alpha r'))) \equiv \pi_{\mathcal{F}}(\sigma_C(\sigma_{C'}(r \times_\alpha r')) \equiv \pi_{\mathcal{F}}(\sigma_{C'}(\sigma_C(r \times_\alpha r')) \equiv \pi_{\mathcal{F}}(\sigma_{C'}(\sigma_C(\sigma_C(r) \times_\alpha \sigma_C(r')))) \equiv \pi_{\mathcal{F}}(\sigma_C(\sigma_{C'}(\sigma_C(r) \times_\alpha \sigma_C(r')))) \equiv \sigma_C(\pi_{\mathcal{F}}(\sigma_{C'}(\sigma_C(r) \times_\alpha \sigma_C(r')))) \equiv \sigma_C(\sigma_C(r) \bowtie_\alpha \sigma_C(r'))$. $\diamond$

Proof of Theorem 12.

- $\sigma_C(r \times_{pc} r') \equiv (\sigma_C(r) \times_{pc} r') \cup (r \times_{pc} \sigma_C(r'))$

  Let $C = L \leq x$. The proofs for $L < x$, $U \leq x$ and $U < x$ are similar.

  • $\sigma_C(r \times_{pc} r') \subseteq (\sigma_C(r) \times_{pc} r') \cup (r \times_{pc} \sigma_C(r'))$.
  Let $at = (d, t, L, U) \in \mathrm{ANN}(\sigma_C(r \times_{pc} r'))$. Then at satisfies C (i.e., $L \leq x$) and $at \in \mathrm{ANN}(r \times_{pc} r')$. By definition of cartesian product there exist two annotated tuples $at' = (d, t, L', U') \in \mathrm{ANN}(r)$ and $at'' = (d, t, L'', U'') \in \mathrm{ANN}(r')$ such that $[L, U] = [L', U'] \otimes_{pc} [L'', U''] = [\min(L', L''), \min(U', U'')]$. Two cases are possible.

  (1) $L = L'$ ($L' \leq L''$). In this case $at' = (d, t, L, U')$ and at' satisfies C, therefore $at' \in \mathrm{ANN}(\sigma_C(r))$. But then, from the above, $at \in \mathrm{ANN}(\sigma_C(r) \times_{pc} r')$.

  (2) $L = L''$ ($L'' < L'$). Reasoning analogously to case (1) we get $at \in \mathrm{ANN}(r \times_{pc} \sigma_C(r'))$.

  Combining two cases together we get $at \in \mathrm{ANN}(\sigma_C(r) \times_{pc} r')$ *or* $at \in \mathrm{ANN}(r \times_{pc} \sigma_C(r'))$. But then, $at \in \mathrm{ANN}((\sigma_C(r) \times_{pc} r') \cup (r \times_{pc} \sigma_C(r')))$.

  • $\sigma_C(r \times_{pc} r') \supseteq (\sigma_C(r) \times_{pc} r') \cup (r \times_{pc} \sigma_C(r'))$.
  Let $at = (d, t, L, U) \in \mathrm{ANN}((\sigma_C(r) \times_{pc} r') \cup (r \times_{pc} \sigma_C(r')))$. By definition of multiset union, two cases are possible.

  (1) $at \in \mathrm{ANN}(\sigma_C(r) \times_{pc} r')$. In this case there exist two annotated tuples $at' = (d, t, L', U') \in \mathrm{ANN}(\sigma_C(r))$ and $at'' = (d, t, L'', U'') \in \mathrm{ANN}(r')$ such that $[L, U] = [L', U'] \otimes_{pc} [L'', U''] = [\min(L', L''), \min(U, U'')]$. As $at' \in \mathrm{ANN}(sigma_C(r)$, at satisfies C and therefore $L' \leq x$. But then as $L = \min(L', L'') \leq L'$, we get $L \leq x$, and therefore at satisfies C. Also, as $at' \in \mathrm{ANN}(\sigma_C(r))$ at is also in $\mathrm{ANN}(r)$. But them $at \in \mathrm{ANN}(r \times_{pc} r')$. As we have established that at satisfies C, $at \in \mathrm{ANN}(\sigma_C(r \times_{pc} r'))$.

  (2) $at \in \mathrm{ANN}(r \times_{pc} \sigma_C(r'))$ and $at \notin \mathrm{ANN}(\sigma_C(r) \times_{pc} r')$. In this case there exist two annotated tuples $at' = (d, t, L', U') \in \mathrm{ANN}(\sigma_C(r))$ and $at'' = (d, t, L'', U'') \in \mathrm{ANN}(r')$ such that $[L, U] = [L', U'] \otimes_{pc} [L'', U''] = [\min(L', L''), \min(U, U'')]$. Because $at \notin \mathrm{ANN}(\sigma_C(r) \times_{pc} r')$, $at' \notin \mathrm{ANN}(\sigma_C(r))$ and therefore at' does not satisfy C, i.e., $L' > x$. On the other hand, $at'' \in \mathrm{ANN}(\sigma_C(r'))$ and therefore (i) $at'' \in \mathrm{ANN}(r')$ and (ii) at satisfies C, i.e., $L'' \leq x$. From the latter and $L' > x$ we conclude $L'' < L$ and therefore $L'' = \min(L', L'')$, i.e., $L = L''$. But then $L \leq x$ and at satisfies C. as $at'' \in \mathrm{ANN}(r)$, at will be contained in $\mathrm{ANN}(r \times_{pc} r')$ and as at satisfies C, $at \in \mathrm{ANN}(\sigma_C(r \times_{pc} r'))$. $\diamond$

- $\sigma_C(r \bowtie_{pc} r') \equiv (\sigma_C(r) \bowtie_{pc} r') \cup (r \bowtie_{pc} \sigma_C(r'))$

  Let $\mathcal{F}$ be the list of join attributes of r and r' and let C' be the appropriate join condition. Then,

$\sigma_C(r \bowtie_{pc} r') = \sigma_C(\pi_{\mathcal{F}}(\sigma_{C'}(r \times_{pc} r'))) \equiv \pi_{\mathcal{F}}(\sigma_C(\sigma_{C'}(r \times_{pc} r'))) \equiv \pi_{\mathcal{F}}(\sigma_{C'}(\sigma_C(r \times_{pc} r'))) \equiv \pi_{\mathcal{F}}(\sigma_{C'}((\sigma_C(r) \times_{pc} r') \cup (r \times_{pc} \sigma_C(r')))) \equiv \pi_{\mathcal{F}}(\sigma_{C'}(\sigma_C(r) \times_{pc} r) \cup \sigma_{C'}(r \times_{pc} \sigma_C(r'))) \equiv \pi_{\mathcal{F}}(\sigma_{C'}(\sigma_C(r) \times_{pc} r')) \cup \pi_{\mathcal{F}}(\sigma_{C'}(r \times_{pc} \sigma_C(r'))) = (\sigma_C(r) \bowtie_{pc} r') \cup (r \bowtie_{pc} \sigma_C(r'))$. $\diamond$

Proof of Theorem 13.

1. $\sigma_{L \leq x}(r \times_{in} r') \equiv \sigma_{L \leq x}(r \times_{in} \sigma_{L \leq \frac{x}{MINL(r)}}(r')) \equiv \sigma_{L \leq x}(\sigma_{L \leq \frac{x}{MINL(r')}}(r) \times_{in} \sigma_{L \leq \frac{MINL(r)}{x}}(r')) \equiv \sigma_{L \leq x}(\sigma_{L \leq \frac{x}{MINL(r')}}(r) \times_{in} r')$.
   We show the first equivalence.
   - $\sigma_{L \leq x}(r \times_{in} r') \subseteq \sigma_{L \leq x}(r \times_{in} \sigma_{L \leq \frac{x}{MINL(r)}}(r'))$
   Let $at = (d,t,L,U) \in \mathrm{ANN}(\sigma_{L \leq x}(r \times_{in} r'))$. Then at satisfies the selection condition, i.e. $L \leq x$ and also $at \in \mathrm{ANN}(r \times_{in} r')$.. Then, by definition of cartesian product, there exist two annotated tuples $at' = (d,t,L',U') \in \mathrm{ANN}(r)$ and $at'' = (d,t,L'',U'') \in \mathrm{ANN}(r')$ such that $[L,U] = [L',U'] \otimes_{in} [L'',U''] = [L' \cdot L'', U' \cdot U'']$. As $L \leq x$, $L' \cdot L'' \leq x$ and therefore $L'' \leq \frac{x}{L'}$. As $MINL(r) \leq L'$, $L'' \leq \frac{x}{L'} \leq \frac{x}{MINL(r)}$. But then $at'' \in \mathrm{ANN}(\sigma_{L \leq \frac{x}{MINL(r)}}(r')$ and therefore, $at \in \mathrm{ANN}(r \times_{in} \sigma_{L \leq \frac{x}{MINL(r)}}(r')$ as at satisfies $L \leq x$, $at \in \mathrm{ANN}(\sigma_{L \leq x}(r \times_{in} \sigma_{L \leq \frac{x}{MINL(r)}}(r')))$.
   - $\sigma_{L \leq x}(r \times_{in} r') \supseteq \sigma_{L \leq x}(r \times_{in} \sigma_{L \leq \frac{x}{MINL(r)}}(r'))$
   Let $at = (d,t,L,U) \in \mathrm{ANN}(\sigma_{L \leq x}(r \times_{in} \sigma_{L \leq \frac{x}{MINL(r)}}(r'))$. Then, at satisfies $L \leq x$ condition and $at \in \mathrm{ANN}((r \times_{in} \sigma_{L \leq \frac{x}{MINL(r)}}(r'))$. By definition of cartesian product, there exist two annotated tuples $at' = (d,t,L',U') \in \mathrm{ANN}(r)$ and $at'' = (d,t,L'',U'') \in \mathrm{ANN}(\sigma_{L \leq \frac{x}{MINL(r)}}(r')$ such that $[L,U] = [L',U'] \otimes_{in} [L'',U''] = [L' \cdot L'', U' \cdot U'']$. Than $at'' \in \mathrm{ANN}(r')$. But then $at \in \mathrm{ANN}(r \times_{in} r')$ and as at satisfies $L \leq x$, $at \in \mathrm{ANN}(\sigma_{L \leq x}(r \times_{in} r'))$. $\diamond$

2. $\sigma_{L<x}(r \times_{in} r') \equiv \sigma_{L<x}(r \times_{in} \sigma_{L<\frac{x}{MINL(r)}}(r')) \equiv \sigma_{L<x}(\sigma_{L<\frac{x}{MINL(r')}}(r) \times_{in} \sigma_{L<\frac{x}{MINL(r)}}(r')) \equiv \sigma_{L<x}(\sigma_{L<\frac{x}{MINL(r')}}(r) \times_{in} r')$.
   We show the second equivalence.
   - $\sigma_{L<x}(r \times_{in} r') \subseteq \sigma_{L<x}(\sigma_{L<\frac{x}{MINL(r')}}(r) \times_{in} \sigma_{L<\frac{x}{MINL(r)}}(r'))$.
   Let $at = (d,t,L,U) \in \mathrm{ANN}(\sigma_{L \leq x}(r \times_{in} r'))$. Then at satisfies the selection condition, i.e. $L \leq x$ and also $at \in \mathrm{ANN}(r \times_{in} r')$.. Then, by definition of cartesian product, there exist two annotated tuples $at' = (d,t,L',U') \in \mathrm{ANN}(r)$ and $at'' = (d,t,L'',U'') \in \mathrm{ANN}(r')$ such that $[L,U] = [L',U'] \otimes_{in} [L'',U''] = [L' \cdot L'', U' \cdot U'']$. As $L \leq x$, $L' \cdot L'' \leq x$ and therefore $L'' \leq \frac{x}{L'}$. As $MINL(r) \leq L'$, $L'' \leq \frac{x}{L'} \leq \frac{x}{MINL(r)}$. Reasoning similarly, $L' \leq \frac{x}{L''}$ and $MINL(r') \leq L''$ yield $L' \leq \frac{x}{MINL(r')}$.
   But then $at'' \in \mathrm{ANN}(\sigma_{L \leq \frac{x}{MINL(r)}}(r')$ and $at' \in \mathrm{ANN}(\sigma_{L \leq \frac{x}{MINL(r')}}(r)$. Therefore,
   $at \in \mathrm{ANN}(\sigma_{L \leq \frac{x}{MINL(r')}}(r) \times_{in} \sigma_{L \leq \frac{x}{MINL(r)}}(r'))$. As at satisfies $L \leq x$,
   $at \in \mathrm{ANN}(\sigma_{L \leq x}(\sigma_{L \leq \frac{x}{MINL(r')}}(r) \times_{in} \sigma_{L \leq \frac{x}{MINL(r)}}(r')))$.
   - $\sigma_{L<x}(r \times_{in} r') \supseteq \sigma_{L<x}(\sigma_{L<\frac{x}{MINL(r')}}(r) \times_{in} \sigma_{L<\frac{x}{MINL(r)}}(r'))$.
   Let $at = (d,t,L,U) \in \mathrm{ANN}(\sigma_{L \leq x}(\sigma_{L<\frac{x}{MINL(r')}}(r) \times_{in} \sigma_{L \leq \frac{x}{MINL(r)}}(r'))$. Then, at satisfies $L \leq x$ condition and $at \in \mathrm{ANN}((\sigma_{L<\frac{x}{MINL(r')}}(r) \times_{in} \sigma_{L \leq \frac{x}{MINL(r)}}(r'))$. By definition of cartesian product, there exist two annotated tuples $at' = (d,t,L',U') \in \mathrm{ANN}(\sigma_{L<\frac{x}{MINL(r')}}(r))$ and $at'' =$

$(d,t,L'',U'') \in \mathrm{ANN}(\sigma_{L \leq \frac{x}{MINL(r)}}(r')$ such that $[L,U] = [L',U'] \otimes_{in} [L'',U''] = [L' \cdot L'', U' \cdot U'']$. Than $at' \in \mathrm{ANN}(r)$ and $at'' \in \mathrm{ANN}(r')$. But then $at \in \mathrm{ANN}(r \times_{in} r')$ and as at satisfies $L \leq x$, $at \in \mathrm{ANN}(\sigma_{L \leq x}(r \times_{in} r'))$. $\diamond$

3. $\sigma_{U \leq x}(r \times_{in} r') \equiv \sigma_{U \leq x}(r \times_{in} \sigma_{U \leq \frac{x}{MINU(r)}}(r')) \equiv \sigma_{U \leq x}(\sigma_{U \leq \frac{x}{MINU(r')}}(r) \times_{in} \sigma_{U \leq \frac{x}{MINU(r)}}(r')) \equiv \sigma_{U \leq x}(\sigma_{U \leq \frac{x}{MINU(r')}}(r) \times_{in} r')$.
   We prove the third equivalence.
   - $\sigma_{L \leq x}(r \times_{in} r') \subseteq \sigma_{L \leq x}(\sigma_{L \leq \frac{x}{MINL(r')}}(r) \times_{in} r')$
   Let $at = (d,t,L,U) \in \mathrm{ANN}(\sigma_{L \leq x}(r \times_{in} r'))$. Then at satisfies the selection condition, i.e. $L \leq x$ and also $at \in \mathrm{ANN}(r \times_{in} r')$.. Then, by definition of cartesian product, there exist two annotated tuples $at' = (d,t,L',U') \in \mathrm{ANN}(r)$ and $at'' = (d,t,L'',U'') \in \mathrm{ANN}(r')$ such that $[L,U] = [L',U'] \otimes_{in} [L'',U''] = [L' \cdot L'', U' \cdot U'']$. As $L \leq x$, $L' \cdot L'' \leq x$ and therefore $L' \leq \frac{x}{L''}$. As $MINL(r') \leq L''$, $L' \leq \frac{x}{L'} \leq \frac{x}{MINL(r')}$. But then $at' \in \mathrm{ANN}(\sigma_{L \leq \frac{x}{MINL(r')}}(r)$ and therefore, $at \in \mathrm{ANN}(\sigma_{L \leq \frac{x}{MINL(r')}}(r) \times_{in} r'$ and, as at satisfies $L \leq x$, $at \in \mathrm{ANN}(\sigma_{L \leq x}(\sigma_{L \leq \frac{x}{MINL(r')}}(r) \times_{in} r'))$.
   - $\sigma_{L \leq x}(r \times_{in} r') \supseteq \sigma_{L \leq x}(r \times_{in} \sigma_{L \leq \frac{x}{MINL(r)}}(r'))$
   Let $at = (d,t,L,U) \in \mathrm{ANN}(\sigma_{L \leq x}(\sigma_{L \leq \frac{x}{MINL(r')}}(r) \times_{in} r')$. Then, at satisfies $L \leq x$ condition and $at \in \mathrm{ANN}(\sigma_{L \leq \frac{x}{MINL(r')}}(r) \times_{in} r')$. By definition of cartesian product, there exist two annotated tuples $at'' = (d,t,L'',U'') \in \mathrm{ANN}(r')$ and $at' = (d,t,L',U') \in \mathrm{ANN}(\sigma_{L \leq \frac{x}{MINL(r')}}(r)$ such that $[L,U] = [L',U'] \otimes_{in} [L'',U''] = [L' \cdot L'', U' \cdot U'']$. Than $at' \in \mathrm{ANN}(r)$. But then $at \in \mathrm{ANN}(r \times_{in} r')$ and as at satisfies $L \leq x$, $at \in \mathrm{ANN}(\sigma_{L \leq x}(r \times_{in} r'))$. $\diamond$

4. $\sigma_{U < x}(r \times_{in} r') \equiv \sigma_{U < x}(r \times_{in} \sigma_{U < \frac{x}{MINU(r)}}(r')) \equiv \sigma_{U < x}(\sigma_{U < \frac{x}{MINU(r')}}(r) \times_{in} \sigma_{U < \frac{x}{MINU(r)}}(r')) \equiv \sigma_{U < x}(\sigma_{U < \frac{x}{MINU(r')}}(r) \times_{in} r')$.
   Proof of the first equivalence is analogous to the proof in part 1 of this theorem. Proof of the second equivalence is analogous to the proof in part 2 of this theorem. Proof of the third equivalence is analogous to the proof in part 3 of this theorem. $\diamond$

Proof of Theorem 14.

Let $at_1 = (d,t,L_1,U_1), \ldots, at_n = (d,t,L_n,U_n)$, be all annotated tuples in $\mathrm{ANN}(r)$ for the pair d,t. Let $\mathcal{F}$ be a list of manifest attributes, and let $d' = \pi_{\mathcal{F}}(d)$. Then, $\mathrm{ANN}(\pi_{\mathcal{F}}(r))$ will contain tuples $at'_1, \ldots, at'_n$ where $at'_i = (d',t,L_i,U_i)$, for $1 \leq i \leq n$. But then, $\mathrm{ANN}(\kappa_{\chi}(\pi_{\mathcal{F}}(r)))$ will contain the tuple $at' = (d,t,L',U')$ where $[L',U'] = \chi([L_1,U_1], \ldots, [L_n,U_n])$.

Now, consider $\mathrm{ANN}(\kappa_{\chi}(r))$. This relation will contain annotated tuple $at = (d,t,L,U)$ where $[L,U] = \chi([L_1,U_1], \ldots, [L_n,U_n]) = [L',U']$. But then, $\mathrm{ANN}(\pi_{\mathcal{F}}(\kappa_{\chi}(r)))$ will contain the tuple $at'' = (d',t,L',U') = at'$. The latter equality proves the theorem. $\diamond$

Proof of Theorem 15.

- Intersection. We show the statement of the theorem for multiset intersection. Then, by Theorem 14, it will also hold for intersection.
  - $\pi_{\mathcal{F}}(r \cap r') \subseteq \pi_{\mathcal{F}}(r) \cap \pi_{\mathcal{F}}(r')$.
  Let $at' = (d',t,L',U') \in \mathrm{ANN}(\pi_{\mathcal{F}}(r \cap r'))$. Then, by definition of projection, there exists an

annotated tuple $at = (d, t, L', U') \in \mathrm{ANN}(r \cap r')$, such that $d' = \pi_{\mathcal{F}}(d)$. As $at \in \mathrm{ANN}(r \cap r')$, either $at \in \mathrm{ANN}(r)$ or $at \in \mathrm{ANN}(r')$. Consider the former case, the latter is symmetric. $at \in \mathrm{ANN}(r \cap r')$ and $at \in \mathrm{ANN}(r)$ implies that there exists an annotated tuple $at'' = (d, t, L'', U'') \in \mathrm{ANN}(r \cap r')$ such that $at'' \in \mathrm{ANN}(r')$. But then, $\mathrm{ANN}(\pi_{\mathcal{F}}(r))$ contains the tuple $at' = (d', t, L', U')$ and $\mathrm{ANN}(\pi_{\mathcal{F}}(r'))$ contains the tuple $at^* = (d', t, L'', U'')$. As at' and at^* are data and time-identical, $\mathrm{ANN}(\pi_{\mathcal{F}}(r) \cap \pi_{\mathcal{F}}(r')$ will contain both at' and at^*.

- $\pi_{\mathcal{F}}(r \cap r') \supseteq \pi_{\mathcal{F}}(r) \cap \pi_{\mathcal{F}}(r')$.

  Let $at' = (d', t, L', U') \in \mathrm{ANN}(\pi_{\mathcal{F}}(r) \cap \pi_{\mathcal{F}}(r')$. Then either $at' \in \mathrm{ANN}(\pi_{\mathcal{F}}(r))$ or $at' \in \mathrm{ANN}(\pi_{\mathcal{F}}(r'))$. Consider the first case, the other case is symmetric.

  $at' \in \mathrm{ANN}(\pi_{\mathcal{F}}(r))$ and $at' = (d', t, L', U') \in \mathrm{ANN}(\pi_{\mathcal{F}}(r) \cap \pi_{\mathcal{F}}(r')$ implies that there exists an annotated tuple $at'' = (d', t, L'', U'') \in \mathrm{ANN}(\pi_{\mathcal{F}}(r) \cap \pi_{\mathcal{F}}(r')$ such that $at'' \in \mathrm{ANN}(\pi_{\mathcal{F}}(r'))$. But then, $\mathrm{ANN}(r)$ contains the tuple $at = (d, t, L', U')$ and $\mathrm{ANN}(r')$ contains the tuple $at^* = (d, t, L'', U'')$ where $d' = \pi_{\mathcal{F}}(d)$. Therefore, as at and at^* are data- and time-identicals, both at and at^* are contained in $\mathrm{ANN}(r \cap r')$. But then, $\mathrm{ANN}(\pi_{\mathcal{F}}(r \cap r'))$ will contain $at' = (d', t, L', U')$ which proves the theorem. $\diamond$

- Union. We prove the theorem for multiset union. Then, by Theorem 14, it will also hold for union.

  - $\pi_{\mathcal{F}}(r \cup r') \subseteq \pi_{\mathcal{F}}(r) \cup \pi_{\mathcal{F}}(r')$.
  Let $at' = (d', t, L', U') \in \mathrm{ANN}(\pi_{\mathcal{F}}(r \cup r'))$. Then, by definition of projection, there exists an annotated tuple $at = (d, t, L', U') \in \mathrm{ANN}(r \cup r')$, such that $d' = \pi_{\mathcal{F}}(d)$. As $at \in \mathrm{ANN}(r \cup r')$, either $at \in \mathrm{ANN}(r)$ or $at \in \mathrm{ANN}(r')$. Consider the former case, the latter is symmetric. As $at \in \mathrm{ANN}(r)$, $\mathrm{ANN}(\pi_{\mathcal{F}}(r))$ contains the tuple $at' = (d', t, L', U')$ and therefore $\mathrm{ANN}(\pi_{\mathcal{F}}(r) \cup \pi_{\mathcal{F}}(r')$ also contains at'.

  - $\pi_{\mathcal{F}}(r \cup r') \supseteq \pi_{\mathcal{F}}(r) \cup \pi_{\mathcal{F}}(r')$.

  Let $at' = (d', t, L', U') \in \mathrm{ANN}(\pi_{\mathcal{F}}(r) \cup \pi_{\mathcal{F}}(r')$. Then either $at' \in \mathrm{ANN}(\pi_{\mathcal{F}}(r))$ or $at' \in \mathrm{ANN}(\pi_{\mathcal{F}}(r'))$. Consider the first case, the other case is symmetric.

  As $at' \in \mathrm{ANN}(\pi_{\mathcal{F}}(r))$, $\mathrm{ANN}(r)$ contains the tuple $at = (d, t, L', U')$ where $d' = \pi_{\mathcal{F}}(d)$. Therefore, as at is contained in $\mathrm{ANN}(r \cup r')$. But then, $\mathrm{ANN}(\pi_{\mathcal{F}}(r \cup r'))$ will contain $at' = (d', t, L', U')$ which proves the theorem. $\diamond$

- Difference.

  - $\pi_{\mathcal{F}}(r - r') \subseteq \pi_{\mathcal{F}}(r) - \pi_{\mathcal{F}}(r')$.
  Let $at' = (d', t, L, U) \in \mathrm{ANN}(\pi_{\mathcal{F}}(r - r'))$. Then, by definition of projection, $\mathrm{ANN}(r - r')$ contains the tuple $at = (d, t, L, U)$ where $d' = \pi_{\mathcal{F}}(d)$. Then, by definition of difference, $at \in \mathrm{ANN}(r)$ and $\mathrm{ANN}(r')$ contains *no* tuple for the pair d, t. But then, $at' = \pi_{\mathcal{F}}(at) \in \mathrm{ANN}(\pi_{\mathcal{F}}(r))$ and $\mathrm{ANN}(\pi_{\mathcal{F}}(r'))$ contains *no* tuple for the pair d', t. Therefore, $at' \in \mathrm{ANN}(\pi_{\mathcal{F}}(r) - \pi_{\mathcal{F}}(r'))$.

  - $\pi_{\mathcal{F}}(r - r') \supseteq \pi_{\mathcal{F}}(r) - \pi_{\mathcal{F}}(r')$.
  Let $at' = (d', t, L, U) \in \mathrm{ANN}(\pi_{\mathcal{F}}(r) - \pi_{\mathcal{F}}(r'))$. Then $at' \in \mathrm{ANN}(\pi_{\mathcal{F}}(r))$ and $\mathrm{ANN}(\pi_{\mathcal{F}}(r'))$

contains no tuple for the pair d', t. But then, ANN(r) contains the tuple $at = (d, t, L, U)$ where $d' = \pi_{\mathcal{F}}(d)$ and ANN(r') contains *no* tuple for the pair d, t. This means that $at \in$ ANN$(r - r')$ and therefore $at' \in$ ANN$(\pi_{\mathcal{F}}(r - r'))$. $\diamond$

Proof of Theorem 16.

- $\pi_{\mathcal{F} \cup \mathcal{F}'}(r \times_\alpha r') \equiv \pi_{\mathcal{F}}(r) \times_\alpha \pi_{\mathcal{F}'}(r')$

  - $\pi_{\mathcal{F} \cup \mathcal{F}'}(r \times_\alpha r') \subseteq \pi_{\mathcal{F}}(r) \times_\alpha \pi_{\mathcal{F}'}(r')$.
  Let $at = (d, t, L, U) \in$ ANN$(\pi_{\mathcal{F} \cup \mathcal{F}'}(r \times_\alpha r'))$. Then, by definition of projection, there exists an annotated tuple $at^* = (d^*, t, L, U) \in$ ANN$(r \times_\alpha r')$ such that $d = \pi_{\mathcal{F} \cup \mathcal{F}'}(d^*)$. By definition of cartesian product, there exist two annotated tuples $at' = (d', t, L', U') \in$ ANN(r) and $at'' = (d'', t, L'', U'') \in$ ANN(r') such that $d* = (d', d'')$ and $[L, U] = [L', U'] \otimes_\alpha [L'', U'']$. But then, $\pi_{\mathcal{F}}(r)$ contains the tuple $at^\star = (d^\star, t, L', U')$ where $d^\star = \pi_{\mathcal{F}}(d')$ and ANN$(\pi_{\mathcal{F}'}(r'))$ contains the tuple $at^{\star\star} = (d^{\star\star}, t, L'', U'')$ where $d^{\star\star} =$ ANN$(\pi_{\mathcal{F}'}(r'))$. But then, ANN$(\pi_{\mathcal{F}}(r) \times_\alpha \pi_{\mathcal{F}'}(r'))$ will contain the tuple $at_1 = ((d^\star, d^{\star\star}), t, L_1, U_1)$ where $[L_1, U_1] = [L', U'] \otimes_\alpha [L'', U'']$. We notice now that $[L_1, U_1] = [L, U]$ and that $(d^\star, d^{\star\star}) = (\pi_{\mathcal{F}}(d'), \pi_{\mathcal{F}'}(d'')) = \pi_(\mathcal{F} \cup \mathcal{F}')(d', d'') = \pi_{\mathcal{F} \cup \mathcal{F}'}(d^*) = d$. Therefore, $at_1 = at$ and $at \in$ ANN$(\pi_{\mathcal{F}}(r) \times_\alpha \pi_{\mathcal{F}'}(r'))$.

  - $\pi_{\mathcal{F} \cup \mathcal{F}'}(r \times_\alpha r') \supseteq \pi_{\mathcal{F}}(r) \times_\alpha \pi_{\mathcal{F}'}(r')$.
  Let $at = (d, t, L, U) \in$ ANN$(\pi_{\mathcal{F}}(r) \times_\alpha \pi_{\mathcal{F}'}(r')$. Then, by definition of cartesian product, there exist annotated tuples $at' = (d', t, L', U') \in$ ANN$(\pi_{\mathcal{F}}(r)$ and $at'' = (d', t, L', U') \in$ ANN$(\pi_{\mathcal{F}'}(r')$, such that $d = (d', d'')$ and $[L, U] = [L', U'] \otimes_\alpha [L'', U'']$. But then, there exists an annotated tuple $at^\star = (d^\star, t, L', U') \in$ ANN(r) and an annotated tuple $at^{\star\star} = (d^{\star\star}, t, L'', U'') \in$ ANN(r') such that $d' = \pi_{\mathcal{F}}(d^\star)$ and $d'' = \pi_{\mathcal{F}'}(d^{\star\star})$. But then, ANN$(r \times_\alpha r')$ contains the tuple $at^* = (d^*, t, L^*, U^*)$ where $d^* = (d^\star, d^{\star\star})$ and $[L^*, U^*] = [L', U'] \otimes_\alpha [L'', U''] = [L, U]$. Then, by definition of projection ANN$(\pi_{\mathcal{F} \cup \mathcal{F}'}(r \times_\alpha r'))$ contains the tuple $at_1 = (d_1, t, L, U)$ where $d_1 = \pi_{\mathcal{F} \cup \mathcal{F}'}(d^*) = \pi_{\mathcal{F} \cup \mathcal{F}'}(d^\star, d^{\star\star}) = (\pi_m athcal F(d^\star), \pi_{\mathcal{F}'}(d^{\star\star}) = (d', d'') = d$. Therefore $at_1 = at$ and $at \in$ ANN$(\pi_{\mathcal{F} \cup \mathcal{F}'}(r \times_\alpha r'))$, which completes the proof. $\diamond$

- $\pi_{\mathcal{F} \cup \mathcal{F}'}(r \bowtie_\alpha r') \equiv \pi_{\mathcal{F}}(r) \bowtie_\alpha \pi_{\mathcal{F}'}(r')$.

  Let $\mathcal{G}$ denote the list of attributes to remain in the join and let C denote the join condition. Recall that $\mathcal{G} \subseteq \mathcal{F}$ and $\mathcal{G} \subseteq \mathcal{F}'$. Then,

  $\pi_{\mathcal{F} \cup \mathcal{F}'}(r \bowtie_\alpha r') \equiv \pi_{\mathcal{F} \cup \mathcal{F}'}(\pi_{\mathcal{G}}(\sigma_C(r \times_\alpha r'))) \equiv \pi_{\mathcal{G}}(\pi_{\mathcal{F} \cup \mathcal{F}'}(\sigma_C(r \times_\alpha r'))) \equiv \pi_{\mathcal{G}}(\sigma_C(\pi_{\mathcal{F} \cup \mathcal{F}'}(r \times_\alpha r'))) \equiv \pi_{\mathcal{G}}(\sigma_C(\pi_{\mathcal{F}}(r) \times_\alpha \pi_{\mathcal{F}'}(r'))) \equiv \pi_{\mathcal{F}}(r) \bowtie_\alpha \pi_{\mathcal{F}'}(r')$. $\diamond$