A Probabilistic Relational Algebra for the Integration of Information Retrieval and Database Systems

Norbert Fuhr, Thomas Rölleke*
University of Dortmund, Germany

Abstract

We present a probabilistic relational algebra (PRA) which is a generalization of standard relational algebra. Here tuples are assigned probabilistic weights giving the probability that a tuple belongs to a relation. Based on intensional semantics, the tuple weights of the result of a PRA expression always confirm to the underlying probabilistic model. We also show for which expressions extensional semantics yields the same results. Furthermore, we discuss complexity issues and indicate possibilities for optimization. With regard to databases, the approach allows for representing imprecise attribute values, whereas for information retrieval, probabilistic document indexing and probabilistic search term weighting can be modelled. As an important extension, we introduce the concept of vague predicates which yields a probabilistic weight instead of a Boolean value, thus allowing for queries with vague selection conditions. So PRA implements uncertainty and vagueness in combination with the relational model.

1 Introduction

Imprecision in databases is a topic which is getting growing attention. New applications of database management systems (DBMS) like technical or scientific databases cannot be handled properly without caring for the intrinsic imprecision of the data. Especially for the integration of information retrieval (IR) and database systems, methods for dealing with uncertainty applied in IR have to be included in an integrated system as well. For this reason, there is a need for data models which can cope with uncertainty.

In the logical view on databases, computing the answer to a query q from a database means to find all objects o for which the logical formula $q \leftarrow o$ is true. If one would take the same approach to document retrieval, then a document d should be retrieved in response to a query if $q \leftarrow d$ can be shown to be true. In fact, this is exactly what Boolean retrieval does. However, since IR has to deal with vagueness and imprecision, this approach is not adequate. For this reason, it is argued in [Rijsbergen 86] that IR should be regarded as an uncertain inference process instead. By using probability theory as basis, Rijsbergen shows that document retrieval is equivalent to computing the probability $P(q \leftarrow d)$ for a document d.

Comparing the two types of inference, one can see that uncertain inference used in IR is just a generalization of the inference mechanism employed in DBMS. So an integration of IR and databases on the logical level seems to be feasible. In order to arrive at a model which can both be implemented and is also applicable in practice, one has to take a data model from the database field and generalize it such that it also comprises probabilistic inference.

In this paper, we present a probabilistic relational model. The basic idea of this model is to assign probabilistic weights to tuples, where the weight of a tuple gives the probability that the tuple belongs to the relation. There is twofold benefit from this approach: First, uncertain data can be stored in the database. Second, the relation computed as an answer to a query reflects the underlying uncertainty in the weights of its tuples; these tuples can be ranked according to decreasing weights, thus yielding the most certain answers on top of the list.

The new model is a generalization of the standard relational model. We regard probabilistic relations as generalizations of ordinary (deterministic) relations. In our model, deterministic relations are treated

*Authors' addresses: University of Dortmund, Informatik VI, 44221 Dortmund, Germany. E-mail: [fuhr,roelleke]@ls6.informatik.uni-dortmund.de. Voice: +49 231 755 2779. Fax: +49 231 755 2386.

as relations allowing only binary (0 or 1) tuple weights, whereas in probabilistic relations, a tuple weight may take any value between 0 and 1. By redefining the basic operators of relational algebra in order to cope with the weights (and their probabilistic interpretation), the laws of relational algebra remain valid. This way, we can use the full power of relational algebra. If a database contains deterministic relations only, our model yields the same result as standard relational algebra. However, in the presence of imprecise data, the answer to a query may contain tuples with nonbinary weights. If ranking is applied, the "certain" tuples will come out first, followed by the uncertain answers.

In order to handle the probabilistic weights in a proper way, our model is based on intensional semantics. In contrast to previous models which are based on extensional semantics, the use of intensional semantics yields a probabilistic relational algebra. (For a further discussion of intensional and extensional semantics for models dealing with uncertainty, see [Pearl 88, pp. 1–14]). For this purpose, tuples of the actual relations in the database represent the basic stochastic events. When a relational algebra operator is applied, each tuple in the result relation is accompanied by an expression that describes the basic events the tuple relates to. Based on this expression, the tuple weight is computed.

In the remainder of this paper, we first describe the motivation that leads to our model. Then we present the probabilistic algebra by giving the basic definitions, followed by a description of the relational operators. In section 4, we discuss the problem of computing the tuple weights, its efficiency and possible optimizations. Section 5 shows how imprecise attribute values are handled by the probabilistic algebra. As an extension to the basic algebra, vague predicates are presented in section 6. Other approaches for imprecision in databases and the integration of IR and database systems are compared with our model in section 7. Finally, we give an outlook on the further development of our model.

2 Motivation

One of the major challenges for incorporating uncertainty in a DBMS is the integration of database and IR systems.

Current commercial IR systems offer very little support for data modelling and query languages of limited expressiveness (and most experimental systems are even worse on these points). As users ask for additional functions, ad-hoc extensions of the query language are developed, without a solid theoretical foundation. So it seems to be attractive to apply models developed in the area of database systems for solving these problems.

On the other hand, many of today's database applications have to cope with text, too. However, even if a DBMS provides some specific functions for text retrieval, these functions do not take into account the intrinsic uncertainty and vagueness of text retrieval. In the field of IR, several types of models have been developed for coping with this problem, and extensive evaluations have demonstrated the feasability of these approaches. Among them, probabilistic IR models offer the advantage of both solid theoretical foundation [Fuhr 92b] and good retrieval performance [Harman 93]. For this reason, a combination of probabilistic IR with a data model like the relational model seems to be a promising goal. But a simple coupling of a standard DBMS with a probabilistic IR system (like e.g. in [Gu et al. 93]) suffers from the fact that the probabilistic weights produced by the IR component cannot be exploited by a DBMS which is based on Boolean logic.

The basic idea of our model is to extend the relational model in such a way that it can handle probabilistic weights required for performing IR. In document indexing, terms are assigned weights w.r.t. the documents in which they occur (see relation INDEX in Figure 1). There is a number of very effective methods for computing these indexing weights automatically (see e.g. [Salton & Buckley 88], [Fuhr & Buckley 91]). The weights are taken into account in retrieval, where the probability of relevance of a document w.r.t. the current query is estimated as a function of the indexing weights of the terms involved. As a simple example, consider a query for documents about databases expressed as $\Pi_{\text{DOCNO}}(\sigma_{\text{TERM='DB'}}(\text{INDEX}))$. The result is shown as relation DB in Figure 1. It is typical for IR queries that the answer is not just a set of objects: Due to the intrinsic uncertainty of IR, the answer should be at least a ranked list of objects. In addition, probabilistic IR models offer an interpretation of the weights that leads to the ranking, namely as estimates of the probability that the object will be judged relevant (on a binary scale) by the user who submitted the query.

INDEX

β	DOCNO	TERM
0.8	1	IR
0.7	1	DB
0.6	2	IR
0.5	3	DB
0.8	3	OOP
0.9	4	IR
0.4	4	AI
0.8	5	DB
0.3	5	OOP

DB

β	DOCNO
0.7	1
0.5	3
0.8	5

Figure 1: Relations representing document indexing and retrieval

Here we have modelled both types of weights as weights assigned to tuples in a relation. Whereas one could imagine other methods of storing indexing weights in a relation (e.g. as additional attribute), this would not be adequate for modelling the weights of answers to a query. As in the standard relational model, where the answer to a query is always a relation again, we would like to have the same property for our new model. For this reason, we generalize the concept of a relation to probabilistic relations, where each tuple is assigned a weight indicating the probability that the tuple belongs to the relation (ordinary relations can be regarded as a special case where only binary weights may be assigned). So any expression in our probabilistic relational algebra yields a probabilistic relation. This approach contrasts with earlier work on probabilistic data models (e.g. [Barbara et al. 92]) or data models for the integration of IR and database systems (e.g. [Schek & Pistor 82]), where weights cannot be assigned to a tuple as a whole, since these models are still based on Boolean logic.

In [Fuhr 93], a probabilistic relational model is described which is based on extensional semantics. A similar model based on Dempster-Shafer theory with extensional semantics is proposed in [Lee 92]. Both approaches suffer from the fact that the result coincides with the underlying theory for simple relational expressions only. As a simple example, consider a query asking for documents either about 'DB' or 'IR', but not about both. If we first would compute a relation IR similar to DB from above, the query could be expressed as (DB–IR) $\cup$ (IR–DB). However, in the definition of the union operator, it is generally assumed that the two arguments are stochastically independent of each other, which is not the case here (the arguments represent disjoint events). In a similar way, the intersection DB $\cap$ IR cannot be computed as DB – (DB – IR). For some special cases, one might think of a simple solution, e.g. an additional parameter indicating the dependence of the two arguments as described in [Lee 92]. However, it is easy to construct examples where the (in)dependence of the relational arguments varies from tuple to tuple (see e.g. Figure 6 below), so we need a different approach to solve more general cases properly. For this reason, we propose a probabilistic algebra which is a generalization of ordinary relational algebra.

3 Description of the algebra

3.1 Basic definitions and assumptions

Let $\bar{R}$ denote an instance of an ordinary relation, and δ a tuple from the corresponding domain. Then we have either $\delta \in \bar{R}$ or $\delta \notin \bar{R}$. In probabilistic relations, we assume that δ is associated with a binary stochastic event η, where η = true, if $\delta \in \bar{R}$, and η = false, otherwise. So we regard the probability $\beta = P(\delta \in \bar{R})$ as additional information belonging to a tuple in a probabilistic relation.

In the following, we will call a probabilistic relation deterministic if for all instances, every probability is either 0 or 1. Otherwise, it is called non-deterministic.

We distinguish between basic events and complex events. The actual database relations (called base relations in the following) only contain basic events. Complex events are Boolean combinations of basic events. They are formed as a by-product of relational operators.

Now we describe the scheme and the possible instances of a probabilistic relation. Let **NAME** denote a set of names which can be used for relations and attributes. A probabilistic relation R is a pair

of relation scheme and relation value, i.e. $R = (S, V)$. Like with ordinary relations, a scheme is a tuple $S = (N, A, D)$ consisting of the relation name $N \in \mathbf{NAME}$, its list of attributes $A = (a_1, \ldots, a_n)$ with $a_i \in \mathbf{NAME}$ for $1 \leq i \leq n$, and the corresponding domains $D = (d_1, \ldots, d_n)$. In the following, let *sch(R)* denote the scheme of R, and *nam(R)*, *attr(R)* and *adom(R)* the three components of *sch(R)*.

The complex domain *dom(R)* of a probabilistic relation is the Cartesian product of the domain of its attributes, i.e. $dom(R) = \bigtimes_{d_i \in adom(R)} d_i$. For a tuple $\delta \in dom(R)$, let $\delta[a_i]$ denote the value for attribute a_i. Furthermore, $\delta[a_1, \ldots, a_k]$ is a shorthand for the tuple $\langle \delta[a_1], \ldots, \delta[a_k] \rangle$. A probabilistic database is a set $\mathcal{R}$ of probabilistic relations with different names.

Since our approach is based on intensional semantics, we need for each tuple in a probabilistic relation information how this tuple depends on the tuples of the base relations and their events. For this purpose, we first need unique identifiers for tuples of base relations. Therefore, we define a function *kv* which creates unique event keys for tuples $\delta \in dom(R)$ as the combination of the relation name and the tuple value: $kv(\delta) := (nam(R), \delta)$. So the corresponding domain of key values for relation R is $kdom(R) = \{nam(R)\} \times dom(R)$.

For certain or impossible events, we need no unique event keys. Instead, we use $\hat{E}$ for certain and E_0 for impossible events. This eases also symbolic manipulation of event expressions (see below). The set of possible event keys is $\mathbf{EK} = \cup_{R \in \mathcal{R}}\ kdom(R) \cup \{\hat{E}, E_0\}$. In relations derived by means of relational operators from the base relations, complex events are represented as Boolean expressions with event keys. The set **EE** of possible event expressions can be defined recursively:

1. $e \in \mathbf{EK} \Rightarrow e \in \mathbf{EE}$.
2. $e \in \mathbf{EE} \Rightarrow (\neg\ e) \in \mathbf{EE}$.
3. $e_1, e_2 \in \mathbf{EE} \Rightarrow (e_1\ \vee\ e_2) \in \mathbf{EE}, (e_1\ \wedge\ e_2) \in \mathbf{EE}$.

These event expressions form a Boolean algebra with $\hat{E}$ as top element and E_0 as bottom element. However, only in the case of databases containing deterministic relations only, they also form a Boolean algebra of truth values. In general, event expressions represent Boolean combinations of events, for which the event probability can be computed (see below). For this reason, assignments and comparisons of event expressions used in the following definitions always relate to expressions formed as Boolean combinations of event keys. In order to ease the readability of the definitions, however, we use Boolean operators $(\wedge, \vee, \neg)$ for event expressions as well as for truth values (of expressions relating to other data elements).

DocTerm

η	β	DocNo	Term
DT(1, IR)	0.9	1	IR
DT(2, DB)	0.7	2	DB
DT(3, IR)	0.8	3	IR
DT(3, DB)	0.5	3	DB
DT(4, AI)	0.8	4	AI

DocAu

η	β	DocNo	AName
DA(1, Bauer)	0.9	1	Bauer
DA(2, Bauer)	0.3	2	Bauer
DA(2, Meier)	0.9	2	Meier
DA(2, Schmidt)	0.8	2	Schmidt
DA(3, Schmidt)	0.7	3	Schmidt
DA(4, Koch)	0.9	4	Koch
DA(4, Bauer)	0.6	4	Bauer

Figure 2: Example probabilistic relations

With these definitions, we can specify the value of a probabilistic relation. Besides a tuple from the complex domain, a single element of the value $V = val(R)$ of a probabilistic relation also comprises a

probabilistic weight and an event expression describing a Boolean combination of basic events. So *val(R)* is a set of triples

$$val(R) \quad = \quad \{(\beta, \eta, \delta) | 0 < \beta \leq 1 \wedge \eta \in \mathbf{EE} \wedge \delta \in dom(R)\}.$$

Here η denotes the event $\delta \in \bar{R}$, and β gives the probability of this event. Additional constraints hold for β and η, depending on the fact whether R is a base relation or a derived relation:

- If $R \in \mathcal{R}$ (R is a base relation), then: if $\beta = 1$ then $\eta = \hat{E}$, otherwise $\eta = kv(\delta)$.
- If $R \notin \mathcal{R}$, $\beta = \varrho(\eta)$.

Here ϱ is a function which maps event expressions onto probabilities. It is defined in section 4. Similar to ordinary relations, tuples t with $t.\beta = 0$ do not belong to the value of a relation.

As an example, Figure 2 shows two probabilistic relations (in our examples, we use abbreviations of the key values), where DocTerm gives the weighted index terms for some documents, and DocAu gives the probability that an author is competent in the subjects described in a paper (e.g. in order to distinguish between primary and other authors).

Now we describe the semantics of a probabilistic database. For this purpose, we use a possible world semantics. For a probabilistic database $\mathcal{R}$, let $\bar{\mathcal{R}}$ denote the corresponding ordinary database containing all tuples of $\mathcal{R}$ with nonzero probability. Furthermore, let $\mathcal{P}(\bar{\mathcal{R}})$ denote the powerset of $\bar{\mathcal{R}}$, i.e. the set of all databases with the same schema as $\bar{\mathcal{R}}$ where all relations are subsets of those of $\bar{\mathcal{R}}$:

$$\mathcal{P}(\bar{\mathcal{R}}) = \{\mathcal{S} | \forall S \in \mathcal{S}\ \exists R \in \bar{\mathcal{R}}\ S \subseteq R\}.$$

Then each possible world W must be a model for an element of $\mathcal{P}(\bar{\mathcal{R}})$. In such a world, a tuple δ is either an element of a relation R, or it is not. Let $\mathcal{W}$ denote the set of all possible worlds in a given interpretation. Then we can define a truth function for all $W \in \mathcal{W}$

$$\tau(R(\delta), W) = \begin{cases} 1 & \text{, if } R(\delta) \text{ is true in world } W \\ 0 & \text{, otherwise} \end{cases}$$

For a tuple t of a probabilistic relation R, the interpretation $\mathcal{I}$ of the event expression $t.\eta$ is given by the truth function for the corresponding data part:

$$\forall W \in \mathcal{W}\ \ \mathcal{I}(t.\eta, W) = \tau(R(t.\delta), W).$$

$\hat{E}$ corresponds to the truth function yielding 1 for every possible world, and E_0 yields 0 in all possible worlds. In addition, we need a probability distribution $P(W)$ over the set of possible worlds. Then the interpretation of the probability $t.\beta$ of a tuple $t \in R$ is

$$\mathcal{I}(t.\beta) = \sum_{W \in \mathcal{W}} P(W) \cdot \tau(R(t.\delta), W).$$

For example, for the probabilistic database shown in figure 2, an interpretation could consist of ten possible worlds with equal probability. Each world contains two (ordinary) relations DocTerm and DocAu, but the relations are different in different worlds; furthermore, no relation will contain more tuples than those shown in figure 2. Then the tuple ⟨1,IR⟩ will be an element of DocTerm in nine of the ten possible worlds.

The operations described in the following form new event expressions as Boolean combinations of the event expressions of their arguments. For our possible world semantics, this means that $R(t.\delta)$ is true at world W iff $\mathcal{I}(t.\eta, W) = 1$, i.e. the Boolean combination of the truth values of the basic events occuring in $t.\eta$ must yield the value true.

3.2 Operations

Given the definitions from above, we can specify the operations of the probabilistic relational algebra. Since the effects on the scheme are the same as in relational algebra, we concentrate on the computation of the value of the result of a PRA operation.

For the definition of the operations, we will use a function $elx(x, R)$ which looks for a tuple r in R with the same data part as x and return its event expression; otherwise, E_0 is returned

$$elx(x, R) := \text{if } \exists r \in val(R)\ (x.\delta = r.\delta) \text{ then } r.\eta \text{ else } E_0$$

First, we describe the two basic set operations. The **union** of relations is defined in relational algebra as $\bar{R} \cup \bar{S} = \{\delta | \delta \in \bar{R} \vee \delta \in \bar{S}\}$. In PRA, this OR-combination of the two events $\delta \in \bar{R}$ and $\delta \in \bar{S}$ applies to the δ part in the same way, and for the event expression, we form the corresponding logical combination of the expressions from the argument relations:

$$\begin{aligned} val(R \cup S) \quad := \quad & \{t | ((\exists r \in val(R)\ (r.\delta = t.\delta)) \vee (\exists s \in val(S)\ (s.\delta = t.\delta))) \wedge \\ & t.\eta = (elx(t, R)\ \vee\ elx(t, S))\} \end{aligned}$$

From the **difference** as defined for relations $\bar{R} - \bar{S} = \{\delta | \delta \in \bar{R} \wedge \delta \notin \bar{S}\}$, we get in a similar way:

$$val(R - S) \quad := \quad \{t | \exists r \in val(R)\ (t.\delta = r.\delta \wedge t.\eta = (r.\eta\ \wedge \neg\ elx(t, S)))\}$$

It should be emphasized that due to the general definition of a probabilistic relation, tuples t with $t.\beta = 0$ do not belong to the result. In ordinary algebra, these tuples are excluded implicitly, but we have to evaluate the event expression of the result in order to detect such tuples

For **selection**, let φ denote the selection formula, and φ' the corresponding formula on tuple level. Then selection is defined in relational algebra as follows: $\sigma_\varphi(\bar{R}) = \{\delta | \delta \in \bar{R} \wedge \varphi'(\delta)\}$. In PRA, this means that we copy all tuples which fulfill the condition φ':

$$val(\sigma_\varphi(R)) := \{t | \exists r \in val(R)\, (t.\delta = r.\delta \wedge \varphi'(r.\delta) \wedge t.\eta = r.\eta)\}$$

Projection has two functions in relational algebra, namely to hide certain attributes and then to remove duplicates. Let $L = (l_1, \ldots, l_m)$ denote the projection list, then we have $\Pi_L(\bar{R}) = \{\delta[l_1, \ldots, l_m] | \delta \in \bar{R}\}$. In the case of a probabilistic relation, we must treat duplicates in a slightly different way: Similar to the union operator where the event expression of the result tuple is the OR-combination of the argument event expressions, for projection the event expression of a result tuple equals the disjunction of the event expressions of the original tuples (i.e., we must compute the probability that at least one of the original tuples is an element of the argument relation). For this purpose, we form the set $S(t)$ which contains all original tuples which are mapped onto the same result tuple t.

$$\begin{aligned} val(\Pi_L(R)) \quad := \quad & \{t | S(t) = \{r | r \in val(R) \wedge r.\delta[l_1, \ldots, l_m] = t.\delta[l_1, \ldots, l_m]\} \wedge S(t) \neq \emptyset \wedge \\ & t.\eta = (\bigvee\nolimits_{r \in S(t)} r.\eta)\} \end{aligned}$$

Finally, the **Cartesian product** is defined in relational algebra as $\bar{R} \times \bar{S} = \{\delta[a_1, \ldots, a_n, b_1, \ldots, b_m] | \delta[a_1, \ldots, a_n] \in \bar{R}\ \wedge \delta[b_1, \ldots, b_m] \in \bar{S}\}$, where $a_1, \ldots, a_n$ are the attributes of R and $b_1, \ldots, b_m$ are the attributes of S. So, for each result tuple, we have the logical AND-combination of the two argument tuples being elements of their corresponding relations:

$$\begin{aligned} val(R \times S) \quad = \quad & \{t | \exists r \in val(R)\ \ \exists s \in val(S)\, (t.\delta[a_1, \ldots, a_n] = r.\delta \wedge \\ & t.\delta[b_1, \ldots, b_m] = s.\delta \wedge t.\eta = (r.\eta\ \wedge\ s.\eta))\} \end{aligned}$$

IR

η	β	DocNo
DT(1, IR)	0.9	1
DT(3, IR)	0.8	3

DB

η	β	DocNo
DT(2, DB)	0.7	2
DT(3, DB)	0.5	3

Figure 3: IR,DB = $\Pi_{\text{DocNo}}(\sigma_{\text{Term='IR','DB'}}(\text{DocTerm}))$

Below, we give some examples for these operations, with the base relations shown in figure 2 (In relation DocAu, authors are given different weights, assuming that e.g. the first author of a paper is

η	β	DocNo
(DT(1, IR) $\vee$ E_0)	0.9	1
(DT(2, DB) $\vee$ E_0)	0.7	2
(DT(3, IR) $\vee$ DT(3, DB))	0.9	3

Figure 4: IR $\cup$ DB

more competent in the subject than his coauthors.). For the computation of the probabilities we assume that all events are independent. Examples of selection and projection are shown in Figure 3, giving the documents dealing with IR or DB, respectively. The union IR $\cup$ DB of these two relations is depicted in Figure 4.

Since we have defined the same basic operations as in standard relational algebra, we can also use the same additional algebraic operations which can be derived from the basic operations. As an application of the join operation, the expression DocAu $\bowtie$ DB as shown in Figure 5 gives authors with documents about DB, which is then projected onto the attribute AName. This is an example of a query which cannot be processed by most of current IR systems.

DocAu $\bowtie$ ($\sigma_{\text{Term='DB'}}$(DocTerm))

η	β	DocNo	Term	AName
(DT(2, DB) $\wedge$ DA(2, B))	0.21	2	DB	Bauer
(DT(2, DB) $\wedge$ DA(2, M))	0.63	2	DB	Meier
(DT(2, DB) $\wedge$ DA(2, S))	0.56	2	DB	Schmidt
(DT(3, DB) $\wedge$ DA(3, S))	0.35	3	DB	Schmidt

Π_{AName}(DocAu $\bowtie$ ($\sigma_{\text{Term='DB'}}$(DocTerm)))

η	β	AName
(DT(2, DB) $\wedge$ DA(2, B))	0.21	Bauer
(DT(2, DB) $\wedge$ DA(2, M))	0.63	Meier
((DT(2, DB) $\wedge$ DA(2, S)) $\vee$ (DT(3, DB) $\wedge$ DA(3, S)))	0.714	Schmidt

Figure 5: Authors writing about DB

As an example for an expression which cannot be evaluated correctly with extensional semantics, consider the following query: we search for authors writing about both IR and DB, but possibly in different papers (see Figure 6). Here the event expression for author Schmidt contains two occurrences of the event DA(3,S); this phenomenon violates the implicit independence assumptions underlying approaches based on extensional semantics.

Π_{AName}(DocAu $\bowtie$ IR) $\cap$ Π_{AName}(DocAu $\bowtie$ DB)

η	β	AName
((DT(1, IR) $\wedge$ DA(1, B)) $\wedge$ (DT(2, DB) $\wedge$ DA(2, B)))	0.1701	Bauer
((DT(3, IR) $\wedge$ DA(3, S)) $\wedge$ ((DT(2, DB) $\wedge$ DA(2, S)) $\vee$ (DT(3, DB) $\wedge$ DA(3, S))))	0.4368	Schmidt

Figure 6: Authors writing both about IR and DB

3.3 PRA as generalization of relational algebra

In contrast to other probabilistic data models (see section 7) which are only loosely related to the relational data model, PRA is a generalization of relational algebra. More precisely,

a) PRA contains relational algebra as a special case, and

b) all equivalences from relational algebra also hold for PRA expressions.

With respect to the first point, we can show that relational algebra corresponds to PRA where all relations are deterministic. For that, let the symbol $\equiv$ denote the equivalence of event expressions (after applying transformations according to Boolean algebra). First, we define the equivalence of ordinary and probabilistic relations: An instance $\bar{R}$ of an ordinary relation and an instance $val(R)$ of a probabilistic relation are equivalent to each other (denoted by $\bar{R} \cong val(R)$), if and only if

1. they have identical schemas, and
2. $(\forall r \in \bar{R}\ \exists t \in val(R)\ (r = t.\delta \wedge t.\eta \equiv \hat{E})) \wedge (\forall t \in val(R)\ \exists r \in \bar{R}\ (r = t.\delta \wedge t.\eta \equiv \hat{E}))$,

Then we can show that the following theorem holds:

Theorem 1 *For any instances $\bar{R}_1, \ldots, \bar{R}_k$ of ordinary relations and any relational algebra expression $\omega(R_1, \ldots, R_k)$, probabilistic relational algebra yields the equivalent result, i.e. if $\bar{R}_1 \cong val(R_1) \wedge \ldots \wedge \bar{R}_k \cong val(R_k)$, then $\omega(\bar{R}_1, \ldots, \bar{R}_k) \cong val(\omega(R_1, \ldots, R_k))$*

The proof of this theorem is given in the appendix.

Concerning the second aspect of generalization (point b) from above), it should be emphasized that in the case of non-deterministic relations, the event expressions formed for each operator yield a probabilistic interpretation of the operator. In [Rölleke 94], it is shown that all equivalences from relational algebra also hold for PRA. This is due to the fact that these equivalences generate equivalent event expressions.

Due to the fact that PRA is a generalization of standard relational algebra, we can also exploit the connection between relational algebra and relational calculus: We use the same transformation process from calculus to algebra but apply the algebraic expression to probabilistic relations. So we have a probabilistic relational calculus which yields probabilistic relations as answers. Thus, PRA forms an implementation of Rijsbergen's view of IR as uncertain inference, since the relational calculus expression for a query can be transformed into an algebra expression and then the answer can be computed. In addition to Rijsbergen's approach, we are able to search not only for documents, but for any item stored in the database.

Finally, since PRA is a generalization of relational algebra, we can apply the well-known methods for query optimization.

4 Computation of event probabilities

The computation of the probability for a given event expression is performed by the function ϱ. Since an event expression is a Boolean combination of basic events, we can apply the well-known sieve formula in order to obtain the probability we are looking for. For that, we first have to transform the event expression into disjunctive normal form (DNF), that is:

$$\eta = K_1 \ \vee \ldots \vee \ K_n,$$

where the K_i are event atoms or conjuncts of event atoms, and an event atom is either an event key or a negated event key (n is the number of conjuncts of the DNF). From Boolean algebra, we know that any Boolean expression can be transformed into DNF. Now we can apply the sieve formula:

$$\varrho(\eta) = P(K_1 \ \vee \ldots \vee \ K_n) = \sum_{i=1}^{n} (-1)^{i-1} \sum_{1 \le j_1 < \ldots < j_i \le n} P(K_{j_1} \ \wedge \ldots \wedge \ K_{j_i}). \qquad (1)$$

For computing the probabilities for the AND-combination of conjuncts, first the combinations containing opposite basic events have to be eliminated and duplicate event keys removed. Then we have to compute the probability of conjuncts of event atoms.

In order to get these probabilities, we need information about the dependence of basic events. If we assume all basic events to be independent of each other, then the probability of a conjunct can be

computed as the product of the probabilities of the event atoms, which in turn are given by the basic relations. Let us call this procedure *standard evaluation* in the following.

On the other hand, if certain events are assumed to be dependent on each other, we need cooccurrence information in order to compute the probabilities of event expressions; this information must be given in additional data structures. So in principle, our approach is able to cope with both independent and dependent basic events. In section 5, we will show how imprecise attribute values can be modelled by assuming disjoint events, and in the case of vague predicates as described in section 6, events are dependent. However, for most applications (like IR), independent events will be sufficient. Furthermore, often the additional parameters required for a dependence model will hardly be available. It should be emphasized that we are considering information systems for vast amounts of data here. In contrast, most probabilistic inference methods are more suitable for expert systems-like applications, where the number of different events is rather limited, and thus it is easier to gather the necessary dependence information. For these reasons, we will mainly discuss the independence case in the following.

For a DNF with n conjuncts, the computation of ϱ takes $O(2^n)$ time. So we should look for possibilities of reducing this computational effort. For that, we have to consider that we introduced the concept of event expressions only in order to cope with possible dependencies of event expressions (caused by identical basic events occurring in these expressions). If no such dependencies exist, then we can compute the event probabilities for the result of an operation directly from the event probabilities of the relational argument(s), without considering the underlying basic events. That is, we treat the result tuples of each operation as basic events in the subsequent operation. Let us call this method *simple evaluation* in the following. So, for PRA expressions where the intensional and extensional semantics approaches would yield the same result, we can eliminate the overhead introduced by the concept of event expressions. For example, the following expression can be evaluated by means of simple evaluation:

$$\Pi_{\mathrm{AName}}(\mathrm{DocAu} \bowtie (\mathrm{IR} \cap \mathrm{DB})),$$

whereas the expression

$$\Pi_{\mathrm{AName}}(\mathrm{DocAu} \bowtie \mathrm{IR}) \cap \Pi_{\mathrm{AName}}(\mathrm{DocAu} \bowtie \mathrm{DB}) \tag{2}$$

requires the more expensive standard evaluation method.

To be more specific, assume that event expressions are only formed according to the definition of the PRA operators, without applying the laws of Boolean algebra. Furthermore, we consider only non-deterministic relations here. Then the following theorem holds:

Theorem 2 *For non-deterministic relations, the result of a PRA expression can be computed by simple evaluation if and only if for every possible tuple of the result relation, each basic event occurs at most once in the event expression of the tuple.*

Proof: First, let us show that simple evaluation yields the correct result in the specified case: If no basic event occurs more than once in an event expression, then the corresponding probability can be computed iteratively, since the arguments of each Boolean operator are independent of each other (Boolean combinations of different (independent) basic events are also independent.). This is exactly what simple evaluation does.

Conversely, suppose that simple evaluation yields correct results, but a basic event occurs more than once in the event expression. Since we start with basic events, there must be a PRA operation where for some result tuple, two or more of the argument tuples forming this result tuple are derived from the same basic event. Thus, the corresponding events are dependent of each other, and simple evaluation yields an incorrect result.□

So the complexity of our approach is the same as that of one based on extensional semantics as long as both approaches yield the same result. Only for queries where extensional semantics produces wrong results, our approach has a higher complexity. However, since the laws of relational algebra hold for our approach, there may be also an equivalent expression for which simple evaluation is applicable. This has to be determined by the query optimizer.

Further simplifications are possible when we know which relations are deterministic. For example, expression 2 can be computed by means of simple evaluation if relation DocAu is deterministic. We are currently investigating the whole area of possible optimizations for the computation of PRA expressions.

DY

η	β	DocNo	*Year*
DY(1,1980)	0.8	1	1980
DY(1,1981)	0.2	1	1981
$\hat{E}$	1.0	2	1990
DY(3,1985)	0.4	3	1985
DY(3,1986)	0.4	3	1986
DY(3,1987)	0.2	3	1987

Figure 7: A probabilistic relation for the imprecise attribute Year

5 Imprecise attribute values

Our basic model of probabilistic relations also can be extended for coping with imprecise attribute values. For that, we model imprecise attribute values as probability distributions over the corresponding domain. As an example, assume that for some documents, the publication year is not known precisely. This fact could be represented by the probabilistic relation DY depicted in figure 7, showing that document 1 was published either in 1980 or 1981, document 2 certainly in 1990 and document 3 in 1985, 1986 or 1987. In contrast to the relations considered so far, DY contains dependent events. Here the events DY(1,1980) and DY(1,1981) are disjoint to each other, in the same way as the three events DY(3,1985), DY(3,1986) and DY(3,1987). On the other hand, it is reasonable to assume that tuples belonging to different document numbers represent independent events. In order to describe the disjointness of events in such a probabilistic relation, we introduce the concept of a *disjointness key*, where two tuples represent disjoint events if they agree on the values of this key (i.e. the attribute DocNo in this example). If the disjointness key is empty, then all tuples of the base relation represent disjoint events.

For this purpose, we extend the scheme of a relation R by a fourth component K to $S = (N, A, D, K)$ with $K \subseteq A$ denoting the disjointness key. Let $K = (k_1, \ldots, k_l)$, then we can define the disjointness set $\mathcal{D}(r)$ for tuple $r \in R$ as follows:

$$\mathcal{D}(r) := \begin{cases} \text{val}(R) & \text{, if } K = \emptyset \\ \{t | t \in R \wedge t.\delta[k_1, \ldots, k_l] \ = r.\delta[k_1, \ldots, k_l]\} & \text{, otherwise} \end{cases}$$

Given this definition, each base relation R must fulfill the following condition:

$$\forall r \in R \ \left(\sum_{t \in \mathcal{D}(r)} t.\beta \right) \leq 1.$$

For the probabilistic relations discussed in the previous sections, there is always $K = A$, i.e. no disjoint tuples. In our examples, the attributes belonging to $A - K$ are printed in italics.

Given the disjointness keys of all relations, the computation of the probabilities for event expressions has to be modified slightly in order to yield correct results in the presence of disjoint events: In the sieve formula (1), AND-combinations of conjuncts containing disjoint events have to be removed (since they represent impossible combinations of events) before computing the probability of conjuncts of event atoms.

For example, the PRA expression $\Pi_{\text{DocNo}}(\sigma_{\text{Year}>1985}(\text{DY}))$ searches for documents published after 1985; as result, we get the event expression (DY(3,1986) $\vee$ DY(3,1987)) for DocNo=3, with a probability of $0.4 + 0.2 = 0.6$.

In comparison with ordinary relations, there is a close relationship between a (non-empty) disjointness key and the key of an ordinary relation. For example, the relation DY from above can be thought of being derived from an ordinary relation BOOK (DocNo, Year, Price, Title) with DocNo as key attribute. When Year is to become an attribute with imprecise values, then Year has to be removed from BOOKS and the new relation DY has to be formed in order to keep relations in first normal form. Since the value of the key attribute(s) determines the values of all other attributes (i.e. the functional dependency DocNo $\rightarrow$ Year in this example), multiple values of a non-key attribute for a single value of a key attribute correspond to disjoint events.

DYP

η	β	DocNo	*Year*	*Price*
DYP(1,1980,20)	0.8	1	1980	20
DYP(1,1981,22)	0.2	1	1981	22
$\hat{E}$	1.0	2	1990	19
DYP(3,1985,15)	0.4	3	1985	15
DYP(3,1986,16)	0.3	3	1986	16
DYP(3,1986,17)	0.1	3	1986	17
DYP(3,1987,17)	0.2	3	1987	17

Figure 8: Imprecise attributes Year and Price

In our possible world semantics, disjoint events represent statements which cannot be true in the same world. Thus, for each possible world, the disjointness key is a key of the relation, since in a single world, the values of the remaining attributes are determined by the value of the disjointness key. However, in another world, the values for the remaining attributes may be different.

In principle, due to the first normal form condition, a separate probabilistic relation has to be formed for each imprecise attribute from the original relation, e.g. DP (DocNo, Price) if Price has also imprecise values. However, this is true only if the values of Price and Year are stochastically independent. Otherwise, both imprecise attributes have to be included in the same relation, as shown in figure 8. Here again the disjointness key is {DocNo}, but now we have several pairs of (Year,Price) values for a single value of DocNo, and these pairs correspond to disjoint events. Obviously, this probabilistic relation cannot be decomposed into two separate relations (even if we ignore the probabilities, there is no lossless join decomposition).

With relation DYP, the query for documents published after 1985 would yield the probability 0.6 for DocNo=3 (as before). Searching for books with a price of at least 17 would give us the probability 0.3 for the same book (and probability 1 for DocNo=1). A selection with the conjunction of the two conditions by means of the expression $\Pi_{\text{DocNo}}(\sigma_{\text{Year}>1985 \wedge \text{Price}\geq 17}(\text{DYP}))$ would yield the event expression (DYP(3,1986,17) $\vee$ DYP(3,1987,18)) with a probability of $0.1 + 0.2 = 0.3$, since the outcomes of the two selection conditions are not independent of each other. In contrast to models with extensional semantics, the (algebraically equivalent) expression $\Pi_{\text{DocNo}}(\sigma_{\text{Year}>1985}(\sigma_{\text{Price}\geq 17}(\text{DYP})))$ would give us the same result.

Q1

η	β	*Term*
Q1(DB)	0.7	DB
Q1(IR)	0.3	IR

Figure 9: Query term weighting

$\Pi_{\text{DocNo}}(\text{Q1} \bowtie \text{DocTerm})$

η	β	DocNo
Q1(IR) $\wedge$ DT(1, IR)	$0.3 \cdot 0.9$	1
Q1(DB) $\wedge$ DT(2, DB)	$0.7 \cdot 0.7$	2
Q1(IR) $\wedge$ DT(3, IR) $\vee$ Q1(DB) $\wedge$ DT(3, DB)	$0.3 \cdot 0.8 + 0.7 \cdot 0.5$	3

Figure 10: Result of query term weighting

Another important application of disjoint events is for query term weighting in text retrieval. So far, we have only considered Boolean combinations of query terms. If we represent the set of query terms as a relation of disjoint events, then we achieve a specific form of query term weighting. For example, a query with the terms DB and IR could be represented as a relation Q1(Term) with the disjointness key {Term} (see figure 9). Then the expression $\Pi_{\text{DocNo}}(\text{Q1} \bowtie \text{DocTerm})$ would yield the result shown in figure 10. Since in Q1 the terms represent disjoint events, but in DocTerm they stand for independent events, the

LT

η	β	A	Price
$\hat{E}$	1.0	1000	900
$\hat{E}$	1.0	1000	950
LT(1000,1000)	0.99	1000	1000
LT(1000,1050)	0.90	1000	1050
LT(1000,1100)	0.60	1000	1100

Figure 11: A probabilistic relation for the vague predicate $\tilde{<}$

probabilities in the result relation are computed as the scalar product of the corresponding weights. This type of query term weighting is also used in the INQUERY system (see e.g. [Haines & Croft 93]).

6 Vague predicates

In interactive information systems, queries often may be vague. For example, a customer looking for PCs with a price less than \$1000 may be willing to pay a little more in certain circumstances. In literature databases, a person searching for relevant publications from the past 3 years may also be interested in a highly relevant paper published 4 years ago. Different approaches have been proposed for this problem, see e.g. [Motro 88], [Prade & Testemale 84], [Zemankova & Kandel 85].

In [Fuhr 90], we have developed a probabilistic interpretation of vague predicates. E.g., in the first example from above, there is a certain probability that the customer will pay a specific price higher than \$1000. This phenomenon can be modelled by means of a probabilistic relation with two attributes containing the values to be compared, and the associated probability that a user formulating a query with this vague predicate will accept a specific pair of values. Figure 11 shows some tuples of the relation for the example from above. Given this relation, it is obvious that we could express a vague selection condition like "Price $\tilde{<}$ 1000" on a relation R by means of the following PRA expression:

$$R \bowtie \Pi_{\text{Price}}(\sigma_{\text{A}=1000}(\text{LT})).$$

It is obvious that the relation corresponding to a vague predicate need not be given explicitly — in many cases, the tuple probability could also be specified as a function of the two values to be compared[1]. In [Fuhr 90], it is shown how this type of function can be derived from empirical data.

Looking more carefully at this example, we see that the independence assumption does not hold for the events in relation LT, since all these events are dependent on each other. However, this fact does not make the whole approach useless. We only run into problems if we get an event expression which contains two different events from this relation. Below, we will show how to cope with this problem. First, however, we will take a closer look at the problem of dependence. In general, a PRA expression might contain more than one vague predicate. So the question is: When are the corresponding events dependent on each other? This problem can be illustrated by means of some examples. Assume that we have a relation PUB(Author, Title, PYear, EYear, Price) about books where PYear denotes the publication year of the first edition and EYear the year of the latest edition. Now consider the following PRA expressions with multiple vague conditions:

1. $\sigma_{\text{Author}\tilde{=}\text{'Smith'} \wedge \text{PYear}\tilde{=}1985}(\text{PUB})$.
2. $\sigma_{\text{PYear}\tilde{>}1990 \wedge \text{Price}\tilde{<}30}(\text{PUB})$
3. $\sigma_{\text{PYear}\tilde{>}1992 \vee \text{PYear}\tilde{<}1985}(\text{PUB})$.
4. $\sigma_{\text{EYear}\tilde{>}1992 \wedge \text{PYear}\tilde{<}1985}(\text{PUB})$.
5. $\sigma_{\text{PYear}\tilde{>}1990 \wedge \text{PYear}\tilde{=}\text{EYear}}(\text{PUB})$.

It is obvious that both in expressions 1 and 2, the events for the two vague predicates are independent of each other. In expression 3, they are dependent, since we have two vague conditions (although with different predicates) for the same attribute. The same holds for expression 5. In 4, we have different

[1] In principle, such a function defines an infinite relation. However, since a user cannot use such a relation explicitely in his PRA expression, no problems with result relations of infinite size occur.

predicates and different attributes, but we would hesitate to say that the events are independent. The reason for this is that the attributes EYear and PYear have the same domain. Since vague predicates bear a lot of semantics, we cannot state the dependence or independence of vague conditions from pure syntactic criteria. Instead, we assume that vague predicates for the same domain are dependent on each other So certain vague predicates can be applied only to attributes of a specific domain. This means that e.g. in the conditions Price$\tilde{<}$30 and PYear$\tilde{<}$1985, $\tilde{<}$ is not the same vague predicate. In order to make this more explicit, we could use the domain as index of the vague predicate, e.g. $\tilde{<}_{\text{Year}}$. On the other hand, we will assume that predicates for different domains are independent of each other.

In the PRA, we must keep track of the dependence information. First, we see that we have certain probabilistic relations with dependent events. Since different vague predicates for the same domain also generate dependent events, this would mean that certain probabilistic relations are also independent of each other in case we use different relations for different predicates. However, in order to simplify the model, we assume that all vague predicates for the same domain are represented by a single probabilistic relation. This can be achieved by using the predicate itself as third attribute. This way, only events within certain probabilistic relations are dependent of each other, whereas events from different relations are always independent.

As an example, for the relation PUB from above, we would have the domain Year with the vague predicates $\tilde{<}$, $\tilde{\leq}$, $\tilde{=}$, $\tilde{\geq}$ and $\tilde{>}$. The corresponding vague predicates would be applicable to the domain Price. For names of persons, different types of string similarity could be provided, e.g. phonetic similarity (by using SOUNDEX or PHONIX codes) or similarity measures based on the number of common trigrams.

Given this type of probabilistic relations, we can transform selections with vague predicates into PRA expressions without vague predicates as follows. First, assume that we have only a simple selection condition, which can be achieved by algebraic transformations. Let DOM(A,B,PR) denote the probabilistic relation for the current domain, where attributes A and B give the two values to be compared, and PR contains the vague predicate.

If the vague selection condition compares two attributes a_i and a_j, we assign the same names to the corresponding attributes of DOM and then use the following transformation:

$$\sigma_{a_i \tilde{\theta} a_j}(\text{R}) = \text{R} \bowtie \Pi_{a_i,a_j}(\sigma_{\text{PR}=\tilde{\theta}}(\text{DOM})).$$

In the case of a comparison with a constant c, we use

$$\sigma_{a_i \tilde{\theta} c}(\text{R}) = \text{R} \bowtie \Pi_{a_i}(\sigma_{\text{PR}=\tilde{\theta} \wedge c=\text{B}}(\text{DOM})).$$

In the following, we will assume that further information about the dependence of events is not available, so we will not be able to compute the probabilities for event expressions containing dependent events. This raises the question which PRA expressions can be processed under this constraint. For this purpose, we introduce the notion of safety: A PRA expression is called *safe* if for all possible values of its arguments, no event expression contains more than one event from a set of dependent events. In this case, standard evaluation yields correct results.

For example, the search for documents with an author name similar to 'Maier' by means of the expression

$$\Pi_{\text{DocNo}}(\sigma_{\text{AName} \,\tilde{=}\, \text{'Maier'}}(\text{DocAu}))$$

is unsafe, since the projection may combine different tuples, where each tuple refers to an event from the relation representing the vague predicate. So the projection operation has to be omitted in order to get a safe expression.

7 Comparison with other approaches

A number of approaches for coping with imprecision in databases have been developed in the past; for a survey, see [IEEE 89], [Motro 90]. A probabilistic relational model with extensional semantics is described in [Cavallo & Pittarelli 87]. In this model, too, probabilistic weights are assigned to tuples, but tuples in the same relation always represent disjoint events. So this model can be regarded as a special case of our approach, and thus the application range is rather limited.

In a certain sense, the PDM model described in [Barbara et al. 92] can be regarded as a further development of the Cavallo and Pittarelli model, in order to overcome some of its disadvantages. This model is similar to a nested relational model model, where tuples of inner relations represent imprecise attribute values. On the other hand, no weights can be assigned to outermost tuples, so this approach lacks a ranking facility. For queries with selection conditions relating to attributes with imprecise values, one can specify a minimum probability with which an atomic condition should be true. However, it is not possible to combine probabilities relating to different conditions. By restricting the set of possible operations, the PDM guarantees that only correct results can be produced; thus many operations possible in PRA cannot be performed within the PDM. The representation of imprecise attribute values in PDM is more intuitive; however, PRA is more expressive by allowing also relations with independent tuples.

Whereas both of these models are based on Bayesian probability theory, the approach presented in [Lee 92] uses Dempster-Shafer theory. Here weights can be assigned to attribute values as well as to tuples as a whole. Imprecise attribute values are represented as sets of values with associated probabilities; thus, relations are not in first normal form. However, since no real nesting of relations is allowed (as in the PDM model), it is not possible to represent imprecise attribute values which are dependent. Furthermore, since this model is based on extensional semantics, it is not a generalization of relational algebra; most equivalences of relational algebra do not hold within this model.

There is also a number of data models based on fuzzy theory (e.g. [Prade & Testemale 84], [Takahashi 93], [Lee & Kim 93]). In principle, probabilistic and fuzzy theory approaches are orthogonal to each other, and the choice of the appropriate theory depends on the actual application. Furthermore, since fuzzy theory is based on extensional semantics, equivalences from relational algebra do not hold for fuzzy relational algebra in general.

The problem of imprecise attribute values in the form of null values or disjunctive information has been discussed extensively in the database literature (see e.g. [Lipski 79] [Vassiliou 79] [Reiter 84] [Imielinski & Lipski 84]); [Imielinski 89] gives a brief survey over this work. As all these approaches are based on two-valued logic, the correct treatment of imprecise values is obvious (e.g. in the proof-theoretic approach [Reiter 84], it follows from the axioms of first-order logic). In [Codd 86], a three-valued logic is used instead, in order to retrieve 'maybe' answers in addition to the correct answers of a query. This model can be regarded as a very simple ranking mechanism. Modifications of this approach are discussed in [Gessert 90] and [Yue 91].

The integration of IR and database systems has been a research topic for several years now. A number of authors have discussed the application of standard relational algebra in IR systems (see e.g. [Desai et al. 87], [Macleod 91]). These papers stress the capability to search for all kinds of objects in the database and the advantages of using a standard database approach, but the approaches lack any ranking facility. In [Blair 88], a model for overcoming this weakness is presented; weights are treated like ordinary attributes here, and ranking is achieved via the `ORDER BY` clause of SQL. However, the combination formulas for weights have to be stated explicitly in the query formulation. In contrast, our approach uses weights implicitly, and the combination formulas are determined by the relational operator. Furthermore, we assume that probabilistic relations are implicitly ranked according to descending probabilities, so no explicit ordering operator is necessary for most queries.

It has been recognized already in [Schek & Pistor 82] that non-first-normal-form (NF^2) relations are suited much better to IR problems than the standard relational model. Its major advantages are easier query formulation for most problems and more efficient processing. On the other hand, there are only minor differences in terms of expressiveness. For this reason, we want to study the general problems with a PRA by using first-normal-form relations before we switch to the NF^2 model.

There are also some approaches for extending text retrieval methods in order to cope with facts, too. If we concentrate on approaches that apply ranking methods at least for texts, then three different methods for the combination of text and fact retrieval can be distinguished:

- Boolean retrieval for facts: The paper [Raghavan et al. 86] describes the combination of probabilistic text retrieval methods with Boolean fact retrieval. Here the fact conditions select a set of objects from the database, followed by a ranking of objects according to the text conditions.
- Non-Boolean retrieval for facts with binary weighting for fact conditions: A simple approach for applying ranking methods to fact conditions as well as to text conditions is to treat fact conditions like index terms in combination with binary indexing. In the SMART system ([Buckley 85]),

different so-called "concept types" can be distinguished; besides text terms, for example dates or names of persons or institutions can be used for retrieval. However, for these attributes, only tests on equality can be performed. In the approach presented in [Saxton & Raghavan 90], a (IR-type) query is a disjunction of conditions, where a single condition can be a Boolean combination of sub-conditions. For fact conditions, the standard predicates and comparison operators can be applied. Each condition in the query is given a weight, and then objects are ranked according to the weighted sum of the conditions they are fulfilling. The paper [Rabitti & Savino 90] describes an advanced system for retrieval of multimedia data (including facts). This system uses probabilistic retrieval for text as well as for facts. Both kinds of conditions can be assigned weights with respect to the query, and text terms also can be given probabilistic index terms weights w.r.t. an object.

- Non-Boolean retrieval for facts with non-binary weighting for fact conditions: In [Fuhr 92a], a model for combining probabilistic text retrieval, vague fact conditions and imprecise data is presented. However, this approach is based on a linear data model only.

To sum up the different efforts for the integration of IR and database systems developed in the past, it can be concluded that approaches based on standard data models do not pay attention to the intrinsic uncertainty and vagueness of IR; on the other hand, approaches aiming to extend IR models for coping with facts, too, offer only a very poor data model and/or a query language with limited expressiveness.

8 Conclusions

In this paper, we have described a probabilistic relational algebra which is a generalization of standard relational algebra. It allows for modelling of different kinds of imprecision in databases. Following the concept of intensional semantics, our approach keeps track of the basic events that contribute to a certain tuple of a relation, and thus all probabilities are computed correctly. In comparison to other approaches with extensional semantics, there is no loss in efficiency for relational expressions where both approaches yield the same result, and there is only an overhead introduced by means of the event expressions when extensional semantics yields incorrect results.

In contrast to other probabilistic data models, PRA is a generalization of relational algebra. Thus, descriptive query languages based on relational calculus can be used as an interface to a system based on PRA. Furthermore, the algebra offers the possibility of query optimizations.

From a theoretical point of view, the independence assumptions used throughout this paper may look rather restrictive. Of course, it is no problem to apply PRA in conjunction with more general assumptions about the dependence of events. However, we think that our choice of independence assumptions is realistic for a broad range of applications. On the other hand, the limitations of our approach (especially for vague predicates) are not a theoretical construct only; they relate directly to practical problems in estimating the stochastical dependence information.

PRA allows for a close integration of IR and database systems, since it includes basic IR methods such as probabilistic document indexing and search term weighting. Through the combination of textual and factual data, PRA supports queries which are not possible in current IR or database systems. For factual data alone, there are powerful mechanisms for modelling imprecise data and vague queries.

It should be emphasized that PRA represents a logical data model. Thus, it makes no assumptions about the underlying physical data model. For example, in an integrated IR and database system, it may be appropriate to store textual and factual data in a different way along with different access paths. The strength of our approach, however, is that it offers the user a unified view of the integrated system, namely a generalization of the relational model.

On the other hand, using the relational model as a standard interface to an integrated IR and database system does not mean that end-users will have to use SQL to query such a system. It is possible to implement different user interfaces which allow for easier query formulation. Then user queries have to be translated into the relational query language offered by the underlying system. Vice versa, the relation computed as answer to the query may be presented to the user in a different form.

References

Barbara, D.; Garcia-Molina, H.; Porter, D. (1992). The Management of Probabilistic Data. *IEEE Transactions on Knowledge and Data Engineering 4(5)*, pages 487–502.

Blair, D. C. (1988). An Extended Relational Document Retrieval Model. *Information Processing and Management 24(3)*, pages 349–371.

Buckley, C. (1985). *Implementation of the SMART Information Retrieval System.* Technical Report 85-686, Department of Computer Science, Cornell University, Ithaca, NY.

Cavallo, R.; Pittarelli, M. (1987). The Theory of Probabilistic Databases. In: *Proceedings of the 13th International Conference on Very Large Databases*, pages 71–81. Morgan Kaufman, Los Altos, Cal.

Codd, E. F. (1986). Missing Information (Applicable and Inapplicable) in Relational Databases. *SIGMOD RECORD 15(4)*, pages 53–78.

Desai, B.; Goyal, P.; Sadri, F. (1987). Non-First Normal Form Universal Relations: An Application to Information Retrieval Systems. *Information Systems 12(1)*, pages 49–55.

Fuhr, N.; Buckley, C. (1991). A Probabilistic Learning Approach for Document Indexing. *ACM Transactions on Information Systems 9(3)*, pages 223–248.

Fuhr, N. (1990). A Probabilistic Framework for Vague Queries and Imprecise Information in Databases. In: McLeod, D.; Sacks-Davis, R.; Schek, H. (eds.): *Proceedings of the 16th International Conference on Very Large Databases*, pages 696–707. Morgan Kaufman, Los Altos, Cal.

Fuhr, N. (1992a). Integration of Probabilistic Fact and Text Retrieval. In: Belkin, N.; Ingwersen, P.; Pejtersen, M. (eds.): *Proceedings of the Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 211–222. ACM, New York.

Fuhr, N. (1992b). Probabilistic Models in Information Retrieval. *The Computer Journal 35(3)*, pages 243–255.

Fuhr, N. (1993). A Probabilistic Relational Model for the Integration of IR and Databases. In: *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 309–17. ACM, New York.

Gessert, G. (1990). Four Valued Logic for Relational Database Systems. *Sigmod record 19(1)*, page 35.

Gu, J.; Thiel, U.; Zhao, J. (1993). Effficient Retrieval of Complex Objects: Query Processing in a hybrid DB and IR System. In: *Proceedings 1. GI-Fachtagung Information Retrieval*, pages 67–81. Universitätsverlag Konstanz, Konstanz.

Haines, D.; Croft, B. (1993). Relevance Feedback and Inference Networks. In: Korfhage, R.; Rasmussen, E.; Willett, P. (eds.): *Proceedings of the Sixteenth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 2–11. ACM, New York.

Harman, D. (1993). Overview of the First Text REtrieval Conference. In: Harman, D. (ed.): *The First Text REtrieval Conference (TREC-1).* National Institute of Standards and Technology Special Publication 500-207, Gaithersburg, Md. 20899.

IEEE. (1989). *IEEE Data Engineering 12(2).* Special Issue on Imprecision in Databases.

Imielinski, T.; Lipski, W. (1984). Incomplete Information in Relational Databases. *Journal of the ACM 31(4)*, pages 761–791.

Imielinski, T. (1989). Incomplete Information in Logical Databases. *IEEE Data Engineering Bulletin 12(2)*, pages 29–40.

Lee, D.; Kim, M. (1993). Accommodating Subjective Vagueness through a Fuzzy Extension to the Relational Data Model. *Information systems 18(6)*, pages 363–374.

Lee, S. (1992). An Extended Relational Database Model For Uncertain And Imprecise Information. In: *Proceedings of the 18th VLDB Conference.* Morgan Kaufman.

Lipski, W. (1979). On Semantic Issues Connected with Incomplete Information Databases. *ACM Transactions on Database Systems 4(3)*, pages 262–296.

Macleod, I. (1991). Text Retrieval and the Relational Model. *Journal of the American Society for Information Science 42(3)*, pages 155–165.

Motro, A. (1988). VAGUE: A User Interface to Relational Databases that Permits Vague Queries. *ACM Transactions on Office Information Systems 6(3)*, pages 187–214.

Motro, A. (1990). Accommodating Imprecision in Database Systems: Issues and Solutions. *Sigmod record 19(4)*, page 69.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufman, San Mateo, Cal.

Prade, H.; Testemale, C. (1984). Generalizing Database Relational Algebra for the Treatment of Incomplete/Uncertain Information and Vague Queries. *Information Science 34*, pages 115–143.

Rabitti, F.; Savino, P. (1990). Retrieval of Multimedia Documents by Imprecise Query Specification. In: Bancilhon, F.; Thanos, C.; Tsichritzis, D. (eds.): *Advances in Database Technology - EDBT '90*, pages 203–218. Springer, Berlin et al.

Raghavan, V.; Saxton, L.; Wong, S.; Ting, S. (1986). A Unified Architecture for the Integration of Data Base Management and Information Retrieval Systems. In: Kugler, H.-J. (ed.): *Information Processing 86*, pages 1049–1054. Elsevier, Amsterdam.

Reiter, R. (1984). Towards a Logical Reconstruction of Relational Database Theory. In: Brodie, M.; Mylopoulos, J.; Schmidt, J. (eds.): *On Conceptual Modelling*, pages 191–233. Springer, New York et al.

van Rijsbergen, C. J. (1986). A Non-Classical Logic for Information Retrieval. *The Computer Journal 29(6)*.

Rölleke, T. (1994). *Equivalences of the Probabilistic Relational Algebra*. Technical report, University of Dortmund, Department of Computer Science.

Salton, G.; Buckley, C. (1988). Term Weighting Approaches in Automatic Text Retrieval. *Information Processing and Management 24(5)*, pages 513–523.

Saxton, L.; Raghavan, V. (1990). Design of an Integrated Information Retrieval / Database Management System. *IEEE Transactions on Knowledge and Data Engineering 2(2)*, pages 210–219.

Schek, H.-J.; Pistor, P. (1982). Data Structures for an Integrated Database Management and Information Retrieval System. In: *Proceedings of the 8th International Conference on Very Large Data Bases*, pages 197–207. Morgan Kaufman, Los Altos, Cal.

Takahashi, Y. (1993). Fuzzy Database Query Languages and Their Relational Completeness Theorem. *IEEE transactions on knowledge and data engineering 5(1)*, page 122.

Vassiliou, Y. (1979). Null Values in Database Management - a Denotational Semantics Approach. In: *Proceedings of the ACM SIGMOD International Conference on the Management of Data*. ACM, New York.

Yue, K. (1991). A More General Model for Handling Missing Information Using a 3- Valued Logic. *SIGMOD record 20(3)*, pages 43–49.

Zemankova, M.; Kandel, A. (1985). Implementing Imprecision in Information Systems. *Information Sciences 37*, pages 107–141.

A Proof of equivalence to relational algebra

In PRA, an instance $\bar{R}$ of an ordinary relation is represented as an instance $val(R)$ of a probabilistic relation where all tuples $r \in val(R)$ have $r.\eta \equiv \hat{E}$. The equivalence with ordinary relational algebra can be shown by regarding the event expressions only: When we perform symbolic manipulations of event expressions, the result can be only $\hat{E}$ or E_0. Tuples with $\eta \equiv E_0$ yield $\beta = 0$ and thus, they are excluded from the result relation.

Now we proof for all operations a) that any result tuple in ordinary algebra gets $\eta \equiv \hat{E}$ in PRA and b) that no other tuple in PRA result relations gets $\eta \equiv \hat{E}$.

Union

$$\begin{aligned} val(R \cup S) \quad := \quad & \{t | ((\exists r \in val(R)\ (r.\delta = t.\delta)) \vee (\exists s \in val(S)\ (s.\delta = t.\delta))) \wedge \\ & t.\eta = (elx(t,R)\ \vee\ elx(t,S))\} \end{aligned}$$

a) In RA, $p \in \bar{R} \cup \bar{S} \Rightarrow (p \in \bar{R} \vee p \in \bar{S})$. Thus, we have in PRA $(\exists r \in val(R) \wedge r.\delta = p \wedge r.\eta \equiv \hat{E}) \vee (\exists s \in val(S) \wedge s.\delta = p \wedge s.\eta \equiv \hat{E}))$. Then we can construct a tuple t with $t.\delta = p$, and $t.\eta = (elx(t,R)\ \vee\ elx(t,S)) \equiv \hat{E}$, thus $t \in val(R \cup S)$.

b) If we have a tuple $t \in val(R \cup S)$ with $t.\eta \equiv \hat{E}$, then due to $t.\eta = (elx(t,R)\ \vee\ elx(t,S))$, we have $\exists r \in val(R)$ with $r.\delta = t.\delta \wedge r.\eta \equiv \hat{E}$ and thus $\exists p \in \bar{R}$ with $p = t.\delta$, or $\exists s \in val(S)$ with $s.\delta = t.\delta \wedge s.\eta \equiv \hat{E}$ and thus $\exists p \in \bar{S}$ with $p = t.\delta$. This, in turn, implies $p \in \bar{R} \cup \bar{S}$.

Difference

$$val(R-S) \quad := \quad \{t | \exists r \in val(R)\ (t.\delta = r.\delta \wedge t.\eta = (r.\eta\ \wedge (\neg\ \epsilon lx(t,S))))\}$$

a) In RA, $p \in \bar{R} - \bar{S} \Rightarrow (p \in \bar{R} \wedge p \notin \bar{S})$. Thus, we have in PRA $(\exists r \in val(R) \wedge r.\delta = p \wedge r.\eta \equiv \hat{E}) \wedge \neg(\exists s \in val(S) \wedge s.\delta = p)$. Then we can construct a tuple t with $t.\delta = p$, and $t.\eta = (r.\eta\ \wedge (\neg\ \epsilon lx(t,S)) \equiv \hat{E}$, thus $t \in val(R-S)$.

b) If we have a tuple $t \in val(R-S)$ with $t.\eta \equiv \hat{E}$, then due to $t.\eta = (r.\eta\ \wedge (\neg\ \epsilon lx(t,S))$, we have $\exists r \in val(R)(r.\delta = t.\delta \wedge r.\eta \equiv \hat{E})$ and thus $\exists p \in \bar{R}$ with $p = t.\delta$, but not $\exists s \in val(S)(s.\delta = t.\delta)$, so $\neg\exists p \in \bar{S}$ with $p = t.\delta$). This, in turn, implies $p \in \bar{R} - \bar{S}$.

Selection

$$val(\sigma_\varphi(R)) := \{t | \exists r \in val(R)\ (t.\delta = r.\delta \wedge t.\eta = r.\eta \wedge \varphi'(r.\delta)\}$$

a) In RA, $p \in \sigma_\varphi(\bar{R}) \Rightarrow p \in \bar{R} \wedge \varphi'(p)$. Thus, we have in PRA $(\exists r \in val(R)(r.\delta = p \wedge r.\eta \equiv \hat{E}))$. Furthermore, $\varphi'(r.\delta)$ is true, and so $r \in val(\sigma_\varphi(R))$.

b) If we have a tuple $t \in val(\sigma_\varphi(R))$ with $t.\eta \equiv \hat{E}$, then $\varphi'(t.\delta)$ must be true, and there must be a tuple $r \in val(R)$ with $r.\delta = t.\delta \wedge r.\eta \equiv \hat{E}$. Thus, $\exists p \in \bar{R}(p = r.\delta)$, which implies $p \in \sigma_\varphi(\bar{R})$.

Projection

$$\begin{aligned} val(\Pi_L(R)) \quad := \quad & \{t | S(t) = \{r | r \in val(R) \wedge r.\delta[l_1,\ldots,l_m] = t.\delta[l_1,\ldots,l_m]\} \wedge S(t) \neq \emptyset \wedge \\ & t.\eta = (\bigvee\nolimits_{r \in S(t)} r.\eta)\} \end{aligned}$$

a) In RA, if $q \in \Pi_L(\bar{R})$, then $\exists p \in \bar{R}(p[l_1,\ldots,l_m] = q)$. Thus, we have in PRA $\exists t \in val(\Pi_L(R))$ with $t.\delta = p[l_1,\ldots,l_m]$, because $\exists r \in val(R)\ (r.\delta = p \wedge r.\eta = \hat{E})$ and $r \in S(t)$. Thus, $S(t) \neq \emptyset$ and $(\bigvee_{r \in S(t)} r.\eta) \equiv \hat{E}$.

b) Assume that $\exists t \in val(\Pi_L(R))$ with $t.\eta \equiv \hat{E}$. This event expression is produced by $(\bigvee_{r \in S(t)} r.\eta)$. Thus, $\exists r \in S(t)$ with $r.\eta \equiv \hat{E} \Rightarrow r \in val(R) \wedge r.\delta[l_1,\ldots,l_m] = t.\delta$. Then we have in RA $\exists p \in \bar{R}\ (p = r.\delta)$, which implies that $t.\delta = p[l_1,\ldots,l_m] \in \Pi_L(\bar{R})$.

Cartesian product

$$\begin{aligned} val(R \times S) \quad = \quad & \{t | \exists r \in val(R)\ \ \exists s \in val(S)\ (t.\delta[a_1,\ldots,a_n] = r.\delta \wedge \\ & t.\delta[b_1,\ldots,b_m] = s.\delta \wedge t.\eta = (r.\eta\ \wedge\ s.\eta))\} \end{aligned}$$

a) In RA, we have $u \in \bar{R} \times \bar{S} \Rightarrow \exists p \in \bar{R}(u[a_1,\ldots,a_n] = p) \wedge \exists q \in \bar{S}(u[b_1,\ldots,b_m] = q)$. Then, in PRA, $\exists r \in val(R)\ (r.\delta = p \wedge r.\eta \equiv \hat{E}) \wedge \exists s \in val(S)\ (s.\delta = q \wedge s.\eta \equiv \hat{E})$. Then, we can construct a tuple t with $t.\delta[a_1,\ldots,a_n] = r.\delta$, $t.\delta[b_1,\ldots,b_m] = s.\delta$, thus $t.\delta = u$. Since $t.\eta = (r.\eta\ \wedge\ s.\eta) \equiv \hat{E}$, $t \in val(R \times S)$.

b)Assume that we have in PRA a tuple $t \in val(R \times S)$ with $t.\eta \equiv \hat{E}$. Then there must be tuples $r \in val(R)$ with $t.\delta[a_1,\ldots,a_n] = r.\delta$ and $s \in val(S)$ with $t.\delta[b_1,\ldots,b_m] = s.\delta$. Furthermore, in order to yield $\hat{E} \equiv t.\eta = (r.\eta \wedge s.\eta)$, $r.\eta \equiv \hat{E}$ and $s.\eta \equiv \hat{E}$. Thus, we have in RA $\exists p \in \bar{R}\ p = r.\delta$ and $\exists q \in \bar{S}\ q = s.\delta$. This, in turn, implies that there is a tuple u with $u[a_1,\ldots,a_n] = p \wedge u[b_1,\ldots,b_m] = q \Rightarrow u \in \bar{R} \times \bar{S}$. However, since $u = t.\delta$, there is always a corresponding tuple in RA for any tuple in $val(R \times S)$

Now we have shown that for the five basic operations, RA and PRA are equivalent. Since all other operations are defined based on the basic operations, we can conclude that PRA and RA are equivalent. □