

A Probabilistic NF2 Relational Algebra for Integrated Information Retrieval and Database Systems

Norbert Fuhr, Thomas Rölleke

June 18, 1996

Abstract

The integration of information retrieval (IR) and database systems requires a data model which allows for modelling documents as entities, representing uncertainty and vagueness and performing uncertain inference. For this purpose, we present a probabilistic data model based on relations in non-first-normal-form (NF2). Here, tuples are assigned probabilistic weights giving the probability that a tuple belongs to a relation. Thus, the set of weighted index terms of a document are represented as a probabilistic subrelation. In a similar way, imprecise attribute values are modelled as a set-valued attribute. We redefine the relational operators for this type of relations such that the result of each operator is again a probabilistic NF2 relation, where the weight of a tuple gives the probability that this tuple belongs to the result. By ordering the tuples according to decreasing probabilities, the model yields a ranking of answers like in most IR models. This effect also can be used for typical database queries involving imprecise attribute values as well as for combinations of database and IR queries.

1 Introduction

The integration of information retrieval (IR) and databases (DB) is an issue which gets growing attention both in research and applications. In scientific as well as in commercial environments, most documents are available now in electronic form, and so there is an interest in storing them in an IR system such that content-based access is possible. However, especially in the commercial area, storing factual and textual data in separate system is not a reasonable solution, since there are often close relationships between the two different types of data, which cannot be modelled by using two specialized systems. For this reason, there is a clear trend towards integrated systems. For example, several (commercial) relational DBMS are offering text retrieval capabilities now. From a theoretical point of view, however, these systems offer very poor IR functions, since they are based on Boolean retrieval. The poor retrieval quality and the user-unfriendliness of this retrieval method are well-known (at least in the IR field, see e.g. [Salton et al. 83]).

So there is a need for developing integrated IR-DB systems based on a better theoretical foundation. The clue to an integrated system lies in the development of a data model which supports both IR and DB needs. Since hardly any IR system is based on an explicit data model, we start with a such a model from the DB field and extend it according to the special needs of IR.

From the IR point of view, a data model should offer the following features:

1. Typical IR objects — i.e. documents — should be modelled as entities. As several authors have pointed out, the relational model (with relations in first normal form) is not appropriate for this purpose, since a document is represented as a collection of tuples spread across several relations; this is due to the fact that some attributes (like e.g. authors, keywords) are set valued. So IR based on the relational model makes query formulation difficult and presentation of answers cumbersome. In contrast, the nested relational model (or NF2 model, for short) allows for set valued attributes, thus a document can be stored as a single tuple in a NF2 relation (see e.g. [Schek & Pistor 82]); this makes queries less complex and results are closer to the user's conceptual model of documents.
2. There should be a means for representing uncertain data and vague queries. Typically, document indexing is uncertain, where the uncertainty is reflected in the document index term weights. A similar problem arises in databases when imprecise attribute values have to be handled. Vice

versa, the vagueness of IR queries is expressed in the search term weights. Obviously, handling uncertainty factors as additional attribute values is not an appropriate solution, since this would leave the interpretation of these values to the application programs. So uncertainty should be included in the semantics of the data model.

3. The operations of the data model should be based on uncertain inference. An essential feature of IR systems is ranking of query results according to increasing values of uncertainty. This can be achieved only if the data model allows for inferring new uncertainty values, and if there also is a means for representing a ranked result within the model.

In order to satisfy these requirements, we propose a probabilistic NF2 model for integrated IR and DB systems. Our approach aims at combining the full expressiveness of a nested relational algebra with probability theory. For this purpose, we generalize the concept of a relation to a probabilistic relation, where each tuple is associated with a binary stochastic event and its probability of being true. Only if this event has the value true, then the tuple belongs to the underlying (ordinary) relation. We redefine the relational operators for this type of relations, thus getting a generalization of NF2 algebra for ordinary relations. Our probabilistic NF2 (pNF2) model treats deterministic and probabilistic relations in the same way. Document indexing weights, imprecise attribute values and query term weights are represented as relation-valued attributes in the pNF2 structure. In the relation computed as an answer to a query, the tuple weights give the probability that a tuple is a correct answer to the query. If we rank tuples according to decreasing values of this probability, then certain tuples (if there are any) will come out first, followed by tuples which are less certain.

The major benefits of our approach are the following:

- The data model is expressive enough to satisfy both IR and DB needs. Thus, it can handle IR and DB queries equally well, and it offers the new possibility to formulate queries involving both factual and textual data.
- For IR applications, the model allows for new types of queries that cannot be handled by traditional IR systems, namely all queries asking for other items than documents (e.g. for authors writing about certain subjects).
- For DB applications with imprecise data, queries addressing this data will lead to ranked results — in contrast to typical DB queries returning sets of entities.

In the following, after a brief survey on related work, we describe the structure of pNF2 relations. Then, the operators from the NF2 algebra are redefined in order to cope with pNF2 relations. This way, we get the basic operations for a nested relational model for different kinds of imprecise data. In section 7, the computation of the probabilistic tuple weights is described. Finally, we give an outlook on further work.

2 Related work

The integration of IR and database systems has been investigated for several years now. The application of standard relational algebra in IR systems is discussed e.g. in [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.

[Schek & Pistor 82] is the first paper on NF2 relations for IR. It also discusses the handling of indexing weights, but treats them as ordinary attributes. [Desai et al. 87] use NF2 relations in combination with the universal relation model, i.e. the whole database is represented as a single NF2 relation. Thus, the user is shielded from the more complex internal structure of the database, and query formulation is easier. [Jarvelin & Niemi 95] describe a query language for IR databases represented as NF2 relations. This language is less complicated than the algebra presented here, but also less expressive. On the other hand, they also provide aggregation operators, which are not considered yet in our approach. The latter two publications do not consider any kind of uncertainty or indexing weights.

Only a few quantitative approaches for coping with uncertainty in databases have been developed in the past; for a survey, see [IEEE 89] and [Motro 90]. Data models based on fuzzy theory are described

e.g. in ([Prade & Testemale 84], [Takahashi 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.

A probabilistic relational model is described in [Cavallo & Pittarelli 87], where tuples in the same relation always represent disjoint events. The approach presented in [Sadri 91] attributes probabilistic certainty factors to information sources producing tuples; thus, it is possible to compute the certainty of tuples produced by more than one source or of tuples in derived relations. The PDM model described in [Barbara et al. 92] is similar to an NF2 model, where the inner tuples represent imprecise attribute values. However, the algebra does not allow for uncertain inference. With the exception of a few special operators, the model presented in the following is a superset of the PDM model. The approach described in [Lee 92] is based on Dempster-Shafer theory. Here weights can be assigned to attribute values as well as to tuples as a whole, but stochastic dependencies between imprecise attribute values cannot be represented. Besides the limitations in modelling only certain kinds of imprecision, the major weakness of this model is its foundation on extensional semantics. That is, for any relational algebra operator, tuple weights of the result are computed as a function of the argument tuples. This has the effect that certain operations may yield incorrect results (e.g. $A \cap B \neq A - (A - B)$). In [Fuhr & Rölleke 96b], we describe this problem and propose a probabilistic algebra with intensional semantics for relations in first normal form; that is, tuple weights are computed as a function of the probabilities of the underlying basic events, and thus, they are always correct. This algebra is a generalization of ordinary relational algebra in that 1) for deterministic relations, the results of both algebras are equivalent and 2) all equivalences from ordinary relational algebra also hold for PRA expressions. However, there is only poor support for modelling the semantics of imprecise attribute values or typical information retrieval applications.

3 Structure of probabilistic NF2 relations

We start with an informal description of probabilistic relations before we turn to the exact definitions. The basic idea of our approach is to generalize the concept of a relation to a **probabilistic relation**, where a tuple belongs only with a certain probability to the underlying (ordinary) relation. As an example, consider figure 1, where the β columns give these probabilistic weights of tuples.

Let us call the set of relations stored in a database **base relations**. Now we assume that each tuple of a base relation is associated with a binary stochastic variable. If this variable has the value true, then the tuple belongs to the relation; otherwise, it is not an element of the relation. We assume that the probability of being true is given for all variables. In this model, ordinary relations can be regarded as a special case where for all variables, the probability of being true is either 0 or 1. For the probabilistic relations, the relational algebra operators are redefined appropriately. By applying these operators to base relations, new probabilistic relations can be derived. Since we use a model with intensional semantics, a tuple of a derived relation is associated with a Boolean expression containing stochastic variables. If certain assumptions about the (in)dependence of events hold, then the probabilities for tuples of derived relations can be computed correctly from the given probabilities of the base relations.

Our pNF2 model is based to a large extent on the NF2 algebra presented in [Colby 90]. For this reason, we focus on the differences between pNF2 and NF2 and do not describe the effect of pNF2 operations on the relation scheme.

Definition 1 (*Relation scheme*)

Let **NAME** denote a set of names. A relation scheme \mathcal{R} is a pair $R\zeta$, where $R \in \mathbf{NAME}$ is the relation scheme name and ζ is a list of the form (A_1, \dots, A_n) with $A_i \neq A_j$ for $i \neq j$, where each A_i is a relation scheme.

The function $nam(\mathcal{R}) := R$ yields the relation scheme name. The function $attr(\mathcal{R}) := \{A_1, \dots, A_n\}$ yields the set of subrelation schemes. A_i is called atomic attribute if $attr(A_i) = \emptyset$ holds. Otherwise, it is called relation-valued attribute.

In the following, when we refer to a relation scheme \mathcal{R} , we will refer to it by its name R alone.

Figure 1 shows an instance of a probabilistic relation with two atomic attributes (BNO, YEAR) and three relation-valued attributes, where PRICE represents an imprecise attribute value, INDEX is a set of indexing terms with probabilistic weights, and AUTHOR is a set of author names.

Each probabilistic tuple t of a relation consists of three components, namely the tuple $t.\delta$ of attribute values, an event expression $t.\eta$ and the corresponding event probability $t.\beta$. In the case of a base relation

as shown here, the event expression is a key which identifies the probabilistic event associated with the tuple. The interpretation of the probability is described further below. First, we give the definitions for event expressions and its components.

Definition 2 (*Event identifiers*)

Let **EID** denote a set of event identifiers. As special elements, **EID** contains the certain event \hat{E} and the impossible event E_0 .

Event identifiers are used for uniquely identifying stochastic events. For certain events, there is no necessity for distinguishing between two certain events, so all certain events get the identifier \hat{E} . For the symbolic manipulation of event expressions, we also need an identifier E_0 denoting the impossible event (probability zero); however, tuples where the event expression equals E_0 do not belong to an instance of a relation.

In order to support symbolic manipulation of expressions containing EIDs, we add information about the relation to which an event belongs. For events corresponding to tuples of a relation-valued attribute, we also need the identification of the higher-level tuples to which the current tuple belongs.

Definition 3 (*Event keys*)

The set of possible event keys is $\mathbf{EK} = \bigcup_{j=1}^l \times_{i=1}^j (\mathbf{NAME} \times \mathbf{EID})$, where l is the maximum nesting level of all base relations and $\times_{i=1}^j$ denotes the associative cartesian product.

The operators of the pNF2 model generate event expressions which are Boolean expressions containing event keys. Since event keys do not have a truth value, neither do event expressions, they only 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, but not to truth values. 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).

Definition 4 (*Event expressions*)

The set **EE** of possible event expressions is inductively defined as follows:

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. However, only in the case of databases containing exclusively deterministic relations, they also form a Boolean algebra of truth values.

Given these definitions, we can describe the interpretation of the tuple probabilities. We first concentrate on probabilities in base relations.

For tuples in top-level base relations, the meaning of events and the corresponding probabilities is clear: $t.\beta = P(t.\eta = \text{true})$. In the following, we will simply write $P(t.\eta)$ for this probability, and $P(\neg t.\eta)$ or $P(\overline{t.\eta})$ for the probability of the opposite event. For tuples in subrelations, the situation is different. Assume that r is a tuple in a relation-valued attribute R of a tuple t in the relation T . Then the event denoted by $r.\eta$ can be true only if $t.\eta$ is true, since it makes no sense to assume that (with a nonzero probability) there is a value for an attribute, but no corresponding tuple t . This view also leads us to the definition of the probability $r.\beta$ as the conditional probability $P(r.\eta|t.\eta) = P(r.\eta = \text{true}|t.\eta = \text{true})$.

If we have derived relations, then we can use the same interpretations of the event probabilities as with the base relations. The probabilities in derived relations are computed according to the event expressions. This step is described in section 7.

Now we define the domain of a probabilistic relation. For simplicity, we assume that there is a universal domain \mathcal{D} for all atomic attributes. Different domains can be enforced by adding appropriate integrity constraints. Let $\mathcal{P}(X)$ denote the powerset of a set X . Then the domains of relation-valued attributes and top-level relations can be defined recursively:

Definition 5 (*Domain of a relation scheme*)

Let \mathcal{D} denote the universal domain for all atomic attributes. The function dom yields the domain of attributes and relation schemes:

$$\text{dom}(\mathcal{R}) := \begin{cases} \mathcal{D}, & \text{if } R \text{ is an atomic attribute} \\ \mathcal{P}([0, 1] \times \mathbf{EE} \times \times_{A_i \in \text{attr}(\mathcal{R})} \text{dom}(A_i)), & \text{otherwise.} \end{cases}$$

Given these definitions, we can specify the conditions which have to be fulfilled by an instance $val(R)$ of a relation R . As a synonym, we will also use the term “value” for an instance.

Definition 6 (*pNF2 relation*)

A pNF2 relation R is a pair $(\mathcal{R}, \mathcal{V})$, where \mathcal{R} is the relation scheme and $val(R) := \mathcal{V} = \{r | r = \langle \beta, \eta, \delta \rangle\}$ is the relation value.

Furtheron, we define $nam(R) := nam(\mathcal{R})$.

The following conditions must be fulfilled by $val(R)$:

- a) $r.\delta$ is an ordered tuple of the form $\langle a_1, \dots, a_n \rangle$ with $a_i \in dom(A_i)$. If A_i is a relation-valued attribute, then a_i is a relation value.
- b) $\forall s, t \in R (s.\delta = t.\delta \Rightarrow s = t)$
- c) $r.\eta \in \mathbf{EE}$
- d) If R is a top-level relation, then $r.\beta = P(r.\eta)$. Otherwise, R is a relation-valued attribute of a relation T , and $\forall r \in R \exists t \in T (r \in t.\delta[R] \Rightarrow r.\beta = P(r.\eta|t.\eta))$
- e) If R is a base relation, then
 1. $r.\beta > 0$.
 2. $r.\eta$ is a tuple $\langle n, e \rangle$ with
 - $e = \hat{E}$, if $r.\beta = 1$, and $e \in \mathbf{EID} - \{\hat{E}, E_0\}$ otherwise
 - If R is a top-level relation, then $n = nam(R)$. Otherwise, R is the value of a relation-valued attribute R of a relation T , for which there exists a tuple $t \in T$ with $(\beta, \eta, \delta) \in t.\delta[R]$. Then $n = \langle t.\eta, R \rangle$.

Since the scheme of a relation is implicitly given for subrelations, we briefly write R instead of $val(R)$, where the context is clear.

The δ part of a tuple corresponds to a tuple in an ordinary relation. We use the notation $t.\delta[A_i]$ for referring to the value for attribute A_i of tuple $t.\delta$, and $t.\delta[A_1, \dots, A_n]$ is a shorthand for the tuple $\langle t.\delta[A_1], \dots, t.\delta[A_n] \rangle$. The scheme of a relation-valued attribute is defined by the scheme component of the top-level relation. Since the definition of a probabilistic relation as a set of tuples relates to the triples $\langle \beta, \eta, \delta \rangle$, we specify in condition b) that two different tuples must also differ in their δ part. In condition e), we specify the additional conditions for base relations. First, only tuples r with $r.\beta > 0$ belong to a base relation. Tuples with certain events must get the identifier \hat{E} in their event key; this definition plays an important role for testing the equality of tuples (see below) and yields results equivalent to those of an ordinary NF2 algebra.

For the computation of tuple probabilities in derived relations, the dependence resp. independence of events has to be considered. For this purpose, we distinguish the following types of probabilistic relations:

Definition 7 (*Types of probabilistic relations*)

Any probabilistic relation R belongs to one of the following types:

- It is **deterministic**, if $\forall r \in val(R) (r.\eta = \hat{E})$.
- It is **independent**, if it is not deterministic, but $\forall s, t \in val(R)$ with $s \neq t$

$$P(s.\eta \wedge t.\eta) = \begin{cases} P(s.\eta) \cdot P(t.\eta), & \text{if } R \text{ is a top-level-relation} \\ P(s.\eta) \cdot P(t.\eta) \cdot P(u.\eta), & \text{if } R \text{ is a relation-valued} \\ & \text{attribute and } u.\delta[R] = val(R) \end{cases}$$

- It is **disjoint**, if $\forall s, t \in val(R) (s.\eta \neq t.\eta \Rightarrow P(s.\eta \wedge t.\eta) = 0)$.
- Otherwise, it is **dependent**.

In the following, we assume that the type of a probabilistic relation is specified as part of the relational scheme. As an example, consider the relation shown in figure 1. We can denote the type of a relation as a subscript of the relation name, i.e.

$BOOK_{det}(BNO, YEAR, PRICE_{disj}(VAL), INDEX_{ind}(TERM), AUTHOR_{det}(NAME))$

So $BOOK$ is a deterministic relation with a deterministic subrelation $AUTHOR$. $PRICE$ represents an imprecise attribute value, thus it is a disjoint relation. $INDEX$ is an independent relation which gives a set of probabilistic index terms for each book.

An important concept in our pNF2 model is equality of tuples. Several operators of the pNF2 model refer to the definition of tuple equality, e.g. in set operations and projection. The problem is caused by relation-valued attributes. In [Colby 90], it is argued that for deterministic NF2 relations, two types

BOOK												
η	β	BNO	YEAR	PRICE			INDEX			AUTHOR		
				η	β	VAL	η	β	TERM	η	β	NAME
$B\hat{E}$	1.0	1	92	$B\hat{E}P1$	0.6	30	$B\hat{E}I1$	0.9	IR	$B\hat{E}A\hat{E}$	1.0	Smith
				$B\hat{E}P2$	0.4	25	$B\hat{E}I2$	0.8	DB	$B\hat{E}A\hat{E}$	1.0	Jones
$B\hat{E}$	1.0	2	93	$B\hat{E}P3$ 1.0 29			$B\hat{E}I3$ 0.9 AI			$B\hat{E}A\hat{E}$ 1.0 Miller		
$B\hat{E}$	1.0	3	92	$B\hat{E}P4$	0.7	28	$B\hat{E}I4$	0.8	OO P	$B\hat{E}A\hat{E}$ 1.0 Jones		
				$B\hat{E}P5$	0.3	25	$B\hat{E}I5$ 0.9 IR			$B\hat{E}A\hat{E}$ 1.0 Jones		
$B\hat{E}$	1.0	4	90	$B\hat{E}P6$	0.5	32	$B\hat{E}I6$ 0.9 DB			$B\hat{E}A\hat{E}$ 1.0 Jones		
				$B\hat{E}P7$	0.5	28	$B\hat{E}I6$ 0.9 DB			$B\hat{E}A\hat{E}$ 1.0 Jones		

Figure 1: $BOOK_{det}(BNO, YEAR, PRICE_{disj}(VAL), INDEX_{ind}(TERM), AUTHOR_{det}(NAME))$

of tuple equality should be used in parallel, and that there should be variants of each operator for the two definitions: When we have two tuples $\langle 1, \{a, b\} \rangle$ and $\langle 1, \{b, c\} \rangle$ as arguments of a projection or union operator, then we can either treat them as being different, or we can form a single result tuple $\langle 1, \{a, b, c\} \rangle$.

In the pNF2 model, nondeterministic relations cause additional problems. Consider for example two tuples s, t which have the same attribute values for all but one attribute A which is a disjoint relation with a single attribute. Now assume that we have $s.\delta[A] = \{\langle 0.5, a \rangle, \langle 0.5, b \rangle\}$ and $t.\delta[A] = \{\langle 0.4, a \rangle, \langle 0.6, b \rangle\}$ (event expressions are omitted here). Then there is a certain probability that both tuples are equal, namely $P(s.\delta[A]=t.\delta[A]) = 0.5 \cdot 0.4 + 0.5 \cdot 0.6 = 0.5$. However, treating these tuples as equal and carrying on the corresponding event expressions would lead to great problems. For example, a projection operation involving the two tuples from above would yield three different tuples — one for the case that they are equal and two for the opposite case; in general, n probably equal tuples would generate $2^n - 1$ result tuples here. So, we decided that two tuples are equal if and only if they have the same attribute values, i.e. $s.\delta = t.\delta$. Thus, if they contain relation-valued attributes, then also the event expressions have to be considered: Deterministic relations are equal if the δ parts are equal, whereas for nondeterministic relations, also the event expressions must be equal. This definition is a mixture of the two concepts used in object-oriented databases, where we have value equality versus object identity. The relational model and its variants are based on value equality only. In contrast, our approach uses value equality for deterministic data, but some type of object identity (due to the concept of event keys) for nondeterministic data.

4 Operations

Now we can describe the operations of the pNF2 model. We first restrict to the non-nested operations, i.e. operations in which only attributes of the top-level relation can be addressed. Nested operations are described in the following section.

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 returns its event expression; otherwise, E_0 is returned.

Definition 8 $elx(x, R) := \text{if } \exists r \in R \wedge x.\delta = r.\delta \text{ then } r.\eta \text{ else } E_0$

Selection is the most powerful operation. First, we define the structure of a selection condition.

Definition 9 (Selection condition)

ϕ is a selection condition on R if it is one of the following types:

- a) $\phi = \emptyset$.
- b) $\phi = a \theta b$ where either
 - i) a and b are atomic attributes of R , or either a or b is an atomic attribute of R and the other is an atomic value, a and b have compatible domains and $\theta \in \{<, >, \leq, \geq, =, \neq\}$, or
 - ii) a and b are relation-valued attributes of R with compatible domains, or either a or b is a relation-valued attribute and the other is an instance of the attribute and $\theta \in \{\subset, \supset, \subseteq, \supseteq, =, \neq\}$, or
 - iii) b is a relation-valued attribute of R and a is a tuple in some instance of b and $\theta \in \{\in, \notin\}$.

c) ϕ_1 and ϕ_2 are two conditions on R and $\phi = \phi_1 \wedge \phi_2$, or $\phi = \phi_1 \vee \phi_2$, or $\phi = \neg\phi_1$.

Then we can define the selection operation as follows:

Definition 10 (Selection)

If ϕ is a selection condition on R , then $\sigma[\phi](R)$ is defined as follows:

$$\sigma[\phi](R) := \{t \mid \exists r \in R (t.\delta = r.\delta \wedge \phi(r) \neq E_0 \wedge t.\eta = (r.\eta \wedge \phi(r)))\}.$$

Here, the function $\phi(r)$ computes the event expression for a specific tuple r w.r.t. the selection condition ϕ . Only those tuples for which $\phi(r) \neq E_0$ are included in the result, where the event expression of a tuple is formed as the conjunction of the event expression of the argument tuple and $\phi(r)$. In the simple case of the comparison of atomic attribute values, $\phi(r)$ yields either E_0 or \hat{E} . Other expressions are produced if element or set comparisons are involved. For element conditions of the type $e \in S$, $\phi(r)$ yields the event expression of the corresponding tuple from S (or E_0 , respectively). For a subset condition of the form $A \subseteq B$, we use the definition $A \subseteq B \Leftrightarrow \forall x \in DOM (x \in A \Rightarrow x \in B) \Leftrightarrow \forall x \in DOM (\neg(x \in A) \vee (x \in B))$, where DOM denotes the corresponding domain. The other set comparison conditions are expressed in terms of the subset predicate.

Definition 11 (Event expression of condition)

The event expression for a condition is computed according to the cases listed in definition 9:

a) If $\phi = \emptyset$, then $\phi(t) := \hat{E}$.

b) If $\phi = a\theta b$, then

1. if a and b are both attributes, then $\phi(t) := eval(t[a], t[b], \theta)$,
2. if only a is an attribute, then $\phi(t) := eval(t[a], b, \theta)$,
3. if only b is an attribute, then $\phi(t) := eval(a, t[b], \theta)$

The function $eval$ is defined according to the three subcases of b) in definition 9:

- i) If a and b are atomic values, then $eval(a, b, \theta) :=$ if $a\theta b$ then \hat{E} else E_0 .
- ii) If a and b are relation-valued, then

$$eval(a, b, \theta) := \begin{cases} (E_0 \vee \bigwedge_{s \in a} (\neg elx(s, a) \vee elx(s, b))), & \text{if } \theta = \subseteq \\ eval(b, a, \subseteq), & \text{if } \theta = \supseteq \\ (eval(a, b, \subseteq) \wedge eval(b, a, \subseteq)), & \text{if } \theta = = \\ (\neg eval(a, b, =)), & \text{if } \theta = \neq \\ (eval(a, b, \subseteq) \wedge \neg eval(b, a, \subseteq)), & \text{if } \theta = \subset \\ eval(b, a, \subset), & \text{if } \theta = \supset \end{cases}$$

iii) If only b is relation-valued, then

$$eval(a, b, \theta) := \begin{cases} elx(a, b), & \text{if } \theta = \in \\ \neg elx(a, b), & \text{if } \theta = \notin \end{cases}$$

c)

$$\phi(t) := \begin{cases} \phi_1(t) \wedge \phi_2(t) & , \text{if } \phi = \phi_1 \wedge \phi_2 \\ \phi_1(t) \vee \phi_2(t) & , \text{if } \phi = \phi_1 \vee \phi_2 \\ \neg\phi_1(t) & , \text{if } \phi = \neg\phi_1 \end{cases}$$

IRB												
η	β	BNO	YEAR	PRICE			INDEX			AUTHOR		
				η	β	VAL	η	β	TERM	η	β	NAME
$B\hat{E} \wedge B\hat{E}I1$	0.9	1	92	$B\hat{E}P1$	0.6	30	$B\hat{E}I1$	1.0	IR	$B\hat{E}A\hat{E}$	1.0	Smith
				$B\hat{E}P2$	0.4	25	$B\hat{E}I2$	0.8	DB	$B\hat{E}A\hat{E}$	1.0	Jones
$B\hat{E} \wedge B\hat{E}I5$	0.9	3	92	$B\hat{E}P4$	0.7	28	$B\hat{E}I5$	1.0	IR	$B\hat{E}A\hat{E}$	1.0	Jones
				$B\hat{E}P5$	0.3	25						

Figure 2: Selection: $\sigma['IR' \in INDEX](BOOK)$

First we give an example with a single term query, e.g. a query asking for books about IR: $IRB = \sigma['IR' \in INDEX](BOOK)$. Figure 2 shows the result of this query, namely the tuples with BNO=1 and

BNO=3, both with probability 0.9. As an example for a Boolean combination of terms, consider a search for books about IR and DB, expressed as

$$\begin{aligned} \text{IRANDDBB} &= \sigma[\text{'DB'} \in \text{INDEX} \wedge \text{'IR'} \in \text{INDEX}](\text{BOOK}) \\ &= \sigma[\{\text{'DB'}, \text{'IR'}\} \subseteq \text{INDEX}](\text{BOOK}) \end{aligned}$$

In the latter formulation, the constant set represents a deterministic relation.

The following query refers to an imprecise attribute value. Asking for books with a price of exactly \$30 can be expressed as: P30 = $\sigma[30 \in \text{PRICE}](\text{BOOK})$. The result of this query consists of the tuple with BNO=1 only, having a probability of 0.6 .

It should be noted that this type of selection does not allow us to formulate conditions on imprecise attributes other than equality (for which we can use the element operator). For example, asking for books with a price less than \$ 30 can only be performed in combination with the unnest operator (see below). However, a more elegant solution to this problem is provided by the nested operations (see next section).

Our projection operator also allows for renaming and duplicating of attributes.

Definition 12 (*Projection*)

In the projection $\Pi[L](R)$, let L be an argument list of the form $L = A_1:B_1, A_2:B_2, \dots, A_n:B_n$, with $A_i \in \text{attr}(R)$ and $B_i \in \text{NAME}$ for $i = 1, \dots, n$. Then the result is defined as

$$\Pi[L](R) := \{t | \exists s \in R \wedge s.\delta[A_1 \dots A_n] = t.\delta[B_1 \dots B_n] \wedge S(t) = \{r | r \in R \wedge r.\delta[A_1 \dots A_n] = t.\delta[B_1 \dots B_n]\} \wedge t.\eta = \bigvee_{r \in S(t)} r.\eta\}$$

As shorthand notation, the second name in a list element also may be omitted here, thus $\Pi[A_i](R)$ is equivalent to $\Pi[A_i : A_i](R)$. If we get duplicates after eliminating certain attributes (i.e. tuples with equal δ parts), then we collect these duplicates in the set $S(t)$, and we form a single result tuple where the event expression is the OR-combination of the event expressions of the duplicate tuples. For deterministic relations, this yields the same result as non-probabilistic NF2 models; for probabilistic relations, the event expression of the result tuple is formed in a way such that the probability of a tuple belonging to the result relation is computed as the probability that at least one of the argument tuples belongs to its argument relation. As an example of this phenomenon, consider the example in figure 5. On the other hand, the projection $\Pi[\text{YEAR}, \text{PRICE}](\text{BOOK})$ does not yield any such duplicates, since there are no two values of PRICE which are certain and equal (two imprecise values could only be equal if the event keys are equal, too, which is not possible for base relations).

Now we describe the set operations. A tuple is an element of the union of $R \cup S$, if it belongs to R or to S (or both). Thus, we form the disjunction of the corresponding event expressions.

Definition 13 (*Union*)

$$R \cup S := \{t | (\exists r \in R (r.\delta = t.\delta) \vee (\exists s \in S (s.\delta = t.\delta))) \wedge t.\eta = (\text{elx}(t, R) \vee \text{elx}(t, S))\}$$

For the difference, a tuple is an element of $R - S$ if it belongs to R , but not to S .

Definition 14 (*Difference*)

$$R - S := \{t | \exists r \in R (r.\delta = t.\delta \wedge t.\eta = (r.\eta \wedge \neg \text{elx}(r, S)) \wedge t.\eta \neq E_0)\}$$

As in ordinary relational algebra, intersection can be defined as $R \cap S = R - (R - S)$.

Definition 15 (*Cartesian Product*)

$$R \times S := \{t | \exists r \in R \exists s \in S (t.\delta[\text{attr}(R)] = r.\delta \wedge t.\delta[\text{attr}(S)] = s.\delta \wedge t.\eta = (r.\eta \wedge s.\eta))\}$$

A tuple is an element of the Cartesian product $R \times S$ if its first part is an element of R and the second part is an element of S . So we have to form the conjunction of the corresponding event expressions. The cartesian product is important for performing joins. Natural joins in the pNF2 model involving atomic attributes only are defined in the same way as in the relational algebra using the basic operations. If a relation-valued attribute is involved, however, then two tuples may be joined although they have different values for this attribute. Consider, for example, a query searching for books with the same index terms as book number 1, which can be expressed as

$$\begin{aligned} \text{BTNO1} &= \Pi[\text{BNO}, \text{INDEX}, \text{INDEX}](\\ &\quad \sigma[\text{INDEX} = \text{INDEX}](\text{BOOK} \times \Pi[\text{BNO}:\text{BNO}', \text{INDEX}:\text{INDEX}](\sigma[\text{BNO}=1](\text{BOOK})))) \end{aligned}$$

The result is depicted in figure 3.¹ The book number 3 belongs with probability 0.164 to the result, since this is the probability that both sets of index terms are either empty or consist of the term 'IR' only. A

¹Here the event expressions are already simplified using the laws of Boolean algebra.

BTNO1								
η	β	BNO	INDEX			INDEX'		
			η	β	TERM	η	β	TERM
$B\hat{E}$	1.0000	1	$B\hat{E}I1$	0.9000	IR	$B\hat{E}I1$	0.9000	IR
			$B\hat{E}I2$	0.8000	DB	$B\hat{E}I2$	0.8000	DB
$B\hat{E} \wedge \neg B\hat{E}I3 \wedge \neg B\hat{E}I4 \wedge \neg B\hat{E}I1 \wedge \neg B\hat{E}I2$	0.0004	2	$B\hat{E}I3$	0.0000	AI	$B\hat{E}I1$	0.0000	IR
			$B\hat{E}I4$	0.0000	OOP	$B\hat{E}I2$	0.0000	DB
$B\hat{E} \wedge \neg B\hat{E}I2 \wedge (B\hat{E}I5 \wedge B\hat{E}I1 \vee \neg B\hat{E}I5 \wedge \neg B\hat{E}I1)$	0.1640	3	$B\hat{E}I5$	0.9878	IR	$B\hat{E}I1$	0.9878	IR
						$B\hat{E}I2$	0.0000	DB
$B\hat{E} \wedge \neg B\hat{E}I1 \wedge (B\hat{E}I6 \wedge B\hat{E}I2 \vee \neg B\hat{E}I6 \wedge \neg B\hat{E}I2)$	0.0740	4	$B\hat{E}I6$	0.9730	DB	$B\hat{E}I1$	0.0000	IR
						$B\hat{E}I2$	0.9730	DB

Figure 3: Join with imprecise attribute values

similar statement holds for BNO=4. Since INDEX and INDEX' may have different values for a result tuple, it would not make sense to allow natural joins involving relation-valued attributes.

Now we turn to the special operators of NF2 algebra, namely unnest and nest. Unnest serves for 'flattening' NF2 tuples. If we unnest the relation-valued attribute A_n of a relation R , then the event expression of a result tuple is formed as the conjunction of the corresponding R - and A_n -tuples. However, we have the additional problem that different argument tuples may yield the same result tuples, e.g. if we unnest a relation containing the tuples $\langle 1, \{a, b\} \rangle$ and $\langle 1, \{b, c\} \rangle$, the result tuple $\langle 1, b \rangle$ is produced twice. In the pNF2 model, this case is handled in the same way as with projection, i.e. forming the OR-combination of the argument event expressions, which are collected in the set $E(t)$ here:

Definition 16 (*Unnest*)

Let R be a relation and $attr(R) = \{A_1, \dots, A_n\}$, $A'_{n1}, \dots, A'_{nk} \in \mathbf{NAME}$, $\{A_1, \dots, A_n\} \cap \{A'_{n1}, \dots, A'_{nk}\} = \emptyset$. If A_n is a relation valued-attribute with $attr(A_n) = \{A_{n1}, \dots, A_{nk}\}$, then

$$\mu[A_n : A'_{n1}, \dots, A'_{nk}](R) := \{t | \exists r \in R(t.\delta[A_1, \dots, A_{n-1}] = r.\delta[A_1, \dots, A_{n-1}] \wedge \exists s \in r.\delta[A_n](t.\delta[A'_{n1}, \dots, A'_{nk}] = s.\delta[A_{n1}, \dots, A_{nk}]) \wedge E(t) = \{\varepsilon | \exists u \in R \wedge t.\delta[A_1, \dots, A_{n-1}] = u.\delta[A_1, \dots, A_{n-1}] \wedge \exists v \in u.\delta[A_n] \wedge t.\delta[A'_{n1}, \dots, A'_{nk}] = v.\delta[A_{n1}, \dots, A_{nk}] \wedge \varepsilon = (u.\eta \wedge v.\eta)\} \wedge t.\eta = \bigvee_{\varepsilon \in E(t)} \varepsilon\}.$$

BL30				
η	β	BNO	PRICE	
$B\hat{E} \wedge B\hat{E}P2$	0.4	1	25	
$B\hat{E} \wedge B\hat{E}P3$	1.0	2	29	
$B\hat{E} \wedge B\hat{E}P4$	0.7	3	28	
$B\hat{E} \wedge B\hat{E}P5$	0.3	3	25	
$B\hat{E} \wedge B\hat{E}P7$	0.5	4	28	

Figure 4: Unnest followed by selection

With the unnest operation, we can formulate the query looking for the numbers of books with a price less than 30:

$$BL30 = \sigma[PRICE < 30](\mu[PRICE:PRICE](\Pi[BNO,PRICE](BOOK)))$$

However, in the result we have multiple occurrences of some books (see figure 4), and we do not see all possible prices of a book. These problems can be overcome by using the projection operator for duplicating attributes first, and a final projection on the relevant attributes (see figure 5):

$$BL30A = \Pi[BNO,PRICE](\sigma[PRICES < 30](\mu[PRICE':PRICES](\Pi[BNO,PRICE,PRICE:PRICE'](BOOK))))$$

However, a more elegant solution for this problem is provided by means of nested selection (see next section).

The nest operation generates new relation-valued attributes. Here we adopt the definition from [Schek & Scholl 86], which defines nest in terms of selection and projection.

BL30A					
η	β	BNO	PRICE		
			η	β	VAL
$B\hat{E} \wedge B\hat{E}P2$	0.4	1	$B\hat{E}P1$	0.0	30
			$B\hat{E}P2$	1.0	25
$B\hat{E} \wedge B\hat{E}P3$	1.0	2	$B\hat{E}P3$	1.0	29
$B\hat{E} \wedge (B\hat{E}P4 \vee B\hat{E}P5)$	1.0	3	$B\hat{E}P4$	0.7	28
			$B\hat{E}P5$	0.3	25
$B\hat{E} \wedge B\hat{E}P7$	0.5	4	$B\hat{E}P6$	0.0	32
			$B\hat{E}P7$	1.0	28

Figure 5: Unnest with duplicated attributes, followed by selection

Definition 17 (*Nest*)

Let R be a relation and $attr(R) = \{A_1, \dots, A_n\}$, $A' \in \mathbf{NAME}$, $A' \notin attr(R)$. Let further $A1 := A_1, \dots, A_{n-k}$ and $A2 := A_{n-k+1}, \dots, A_n$. Then $\nu[A_{n-k+1}, \dots, A_n : A'](R)$ abbreviated by $\nu[A2 : A'](R)$ is defined as follows:

$$\nu[A2 : A'](R) := \{t | \exists r \in \Pi[A1](R)(t.\delta[A1] = r.\delta[A1] \wedge t.\delta[A'] = \Pi[A2](\sigma[A1 = r.\delta[A1] \wedge \dots \wedge A_{n-k} = r.\delta[A_{n-k}]](R)) \wedge t.\eta = r.\eta\}.$$

As an example, assume that we have a relation BOOK1 similar to BOOK, but the top-level relation is disjoint with equal probabilities for all tuples. So the event probability of a tuple gives the probability that choosing a book randomly would yield this book. If we now want to see the distribution of book prices for the different years, we combine the projection on YEAR and PRICE from above with an unnest-nest sequence:

$$\text{YEARPRICE} = \nu[\text{PRICE:PRICE}](\mu[\text{PRICE:PRICE}](\Pi[\text{YEAR,PRICE}](\text{BOOK1})))$$

We could apply the same expression to relation BOOK, giving us for each price the probability that we could observe this value in any of the tuples from the corresponding year.

5 Nested operations

Similar to the arguments presented in [Roth et al. 86] for the case of deterministic NF2 relations (in partitioned normal form), one could take the view that there is no need for nested operations, since these can all be expressed by means of the nonnested operations. However, we can show that for nondeterministic relations, certain nested operations yield event expressions different from those produced by the corresponding nonnested expression (see [Fuhr & Rölleke 96a]). So the equivalence only holds for deterministic relations. Besides the problem of expressiveness, we think that especially for imprecise attributes, nested operations provide a more elegant way for formulating expressions involving these attributes.

We can easily extend the operations defined in the previous section to nested operations similar to a nested algebra. For this purpose, we allow **path expressions** as an additional parameter for all operators.

Definition 18 P is a path of R if

1. P is empty, or
2. P is of the form $/R_i P_i$, where R_i is a relation-valued attribute of R and P_i is a path of R_i .

In contrast to the NF2 algebra presented in [Colby 90], we use paths in the same way for all operations. Specifying a path as parameter of an operation means that only the relation-valued attribute specified by the path is modified. As an example, assume that among the index terms of books, we only want to see to what extent they are about IR or DB. This can be expressed by means of a selection with a path (see figure 6):

$$\text{IRDBI} = \sigma[\text{TERM}='IR' \vee \text{TERM}='DB']/\text{INDEX}(\text{BOOK})$$

Due to the fact that book number 2 has an empty set of index terms after applying this expression, there is no possibility for achieving the same result without using path expressions; this is an example

IRDBI												
η	β	BNO	YEAR	PRICE			INDEX			AUTHOR		
				η	β	VAL	η	β	TERM	η	β	NAME
$B\hat{E}$	1.0	1	92	$B\hat{E}P1$	0.6	30	$B\hat{E}I1$	0.9	IR	$B\hat{E}A\hat{E}$	1.0	Smith
				$B\hat{E}P2$	0.4	25	$B\hat{E}I2$	0.8	DB	$B\hat{E}A\hat{E}$	1.0	Jones
$B\hat{E}$	1.0	2	93	$B\hat{E}P3$	1.0	29				$B\hat{E}A\hat{E}$	1.0	Miller
$B\hat{E}$	1.0	3	92	$B\hat{E}P4$	0.7	28	$B\hat{E}I5$	0.9	IR	$B\hat{E}A\hat{E}$	1.0	Jones
				$B\hat{E}P5$	0.3	25						
$B\hat{E}$	1.0	4	90	$B\hat{E}P6$	0.5	32	$B\hat{E}I6$	0.9	DB	$B\hat{E}A\hat{E}$	1.0	Jones
				$B\hat{E}P7$	0.5	28						

Figure 6: Nested selection

of the more general problem that expressions which are equivalent in deterministic NF2 algebra yield different event expressions in pNF2 algebra. For this reason, we define special operators for dealing with paths as additional arguments.

Definition 19 (*Unary operations*)

If $\omega \in \{\Pi, \sigma, \nu, \mu\}$, L is an argument list and P is a path of R , then

- if P is empty, then $\omega[L]P(R) := \omega[L](R)$,
- otherwise P has the form $/R_iP_i$, and

$$\omega[L]P(R) := \{t | \exists r \in R(t.\delta[attr(R) - \{R_i\}] = r.\delta[attr(R) - \{R_i\}] \wedge t.\delta[R_i] = \omega[L]P_i(r.\delta[R_i])) \wedge t.\eta = r.\eta\}.$$

Paths in binary operations can be used to navigate into one of the argument relations — but not into both, since this would give no meaningful results.

Definition 20 (*Binary Operations*)

If $\omega \in \{\cup, \times, -\}$, and P is a path of R , then

- if P is an empty path, then

$$(S)\omega P(R) := (S)\omega(R),$$

$$(R)P\omega(S) := (S)\omega(R),$$
- if P is a path of the form $/R_iP_i$, then

$$(S)\omega P(R) := \{t | \exists r \in R(t.\delta[attr(R) - \{R_i\}] = r.\delta[attr(R) - \{R_i\}] \wedge t.\delta[R_i] = (S)\omega P_i(r.\delta[R_i])) \wedge t.\eta = r.\eta\},$$

$$(R)P\omega(S) := \{t | \exists r \in R(t.\delta[attr(R) - \{R_i\}] = r.\delta[attr(R) - \{R_i\}] \wedge t.\delta[R_i] = (r.\delta[R_i])P_i\omega(S) \wedge t.\eta = r.\eta\}.$$

As an example for a binary operation, consider a query with weighted search terms represented as a relation $Q_{disj}(\text{Term})$ depicted in figure 7.

Q		
η	β	TERM
Q1	0.6	IR
Q2	0.4	DB

Figure 7: Query with weighted search terms $Q_{disj}(\text{TERM})$

Now we can formulate the query

$$\text{IRDBW} = \sigma[\text{INDEX} \neq \emptyset](Q \cap / \text{INDEX}(\text{BOOK})),$$

giving the result depicted in figure 8.

In a similar way, we can express the query for books about IR or DB as

$$\begin{aligned} \text{IRORDBB1} &= \sigma[\text{INDEX} \neq \emptyset](\sigma[\text{TERM} = \text{'IR'} \vee \text{TERM} = \text{'DB'}] / \text{INDEX}(\text{BOOK})) \\ &= \sigma[\text{INDEX} \neq \emptyset](\{\text{'DB'}, \text{'IR'}\} \cap / \text{INDEX}(\text{BOOK})) \end{aligned}$$

(Note that these formulations also affect the attribute INDEX as in figure 6.)

In order to simplify the notation for this type of selections, we introduce so-called selection paths:

IRDBW												
η	β	BNO	YEAR	PRICE			INDEX			AUTHOR		
				η	β	VAL	η	β	TERM	η	β	NAME
$B\hat{E} \wedge (Q1 \wedge B\hat{E}11 \vee Q2 \wedge B\hat{E}12)$	0.86	1	92	$B\hat{E}P1$	0.6	30	$Q1 \wedge B\hat{E}11$	0.63	IR	$B\hat{E}A\hat{E}$	1.0	Smith
				$B\hat{E}P2$	0.4	25	$Q2 \wedge B\hat{E}12$	0.37	DB	$B\hat{E}A\hat{E}$	1.0	Jones
$B\hat{E} \wedge Q1 \wedge B\hat{E}15$	0.54	3	92	$B\hat{E}P4$	0.7	28	$Q1 \wedge B\hat{E}15$	1.0	IR	$B\hat{E}A\hat{E}$	1.0	Jones
				$B\hat{E}P5$	0.3	25						
$B\hat{E} \wedge Q2 \wedge B\hat{E}16$	0.36	4	90	$B\hat{E}P6$	0.5	32	$Q2 \wedge B\hat{E}16$	1.0	DB	$B\hat{E}A\hat{E}$	1.0	Jones
				$B\hat{E}P7$	0.5	28						

Figure 8: Result of query with weighted search terms

Definition 21 (*Selection paths*)

The three types of selection conditions from definition 9 are extended by a fourth type:

- d) ϕ is of the form $/S\phi'$, where S is a relation-valued attribute of the current relation and ϕ' is a condition w.r.t. S

This form of selection condition is evaluated as follows:

Definition 22 (*Extended Selection*)

Definition 11 is extended by the following subcase:

- d) If $\phi = /S\phi'$, then $\phi(t) = E_0 \vee \bigvee_{r \in t.\delta[S]} r.\eta \wedge \phi'(r)$

Here, the disjunction over all $r \in t.\delta[S]$ describes the event that $t.\delta[S]$ is different from the empty set. So selection paths implement an existential quantification over the tuples of the relation-valued attribute named in the path.

As an example, searching for books with a price less than 30 in the relation BOOK can now be performed as

$$BL30A = \sigma[/PRICE(VAL < 30)](BOOK).$$

The query for books about IR or DB (without modifying the attribute INDEX) now can be formulated as

$$\begin{aligned} IRORDBB2 &= \sigma['DB' \in INDEX \vee 'IR' \in INDEX](BOOK) \\ &= \sigma[/INDEX('DB' = TERM \vee 'IR' = TERM)](BOOK.) \end{aligned}$$

Universal quantification can be achieved by negated paths. So a query for books which are only about IR or DB is expressed as

$$IRDBONLY = \sigma[\neg /INDEX('DB' \neq TERM \wedge 'IR' \neq TERM)](BOOK.)$$

6 IR application examples

After having described the pNF2 operators, we now want to demonstrate the feasibility of pNF2 algebra for IR applications. So far, we have concentrated on typical IR queries which could be handled by other IR systems as well. Now we want to show that our algebra is able to perform other tasks which are essential in IR applications, too.

First, we give two examples of retrieval using additional knowledge sources. One such source is a thesaurus. Here we assume that a thesaurus is mapped onto a pNF2 relation THESAURUS(ETERM, RT(TERMS)), where RT gives a set of related terms for the entry term ETERM. Now a search about IR or DB or related topics could be performed by first looking up the set of related terms and then searching for books indexed with this set:

$$\begin{aligned} IRDBTERMS &= \Pi[TERMS](\mu[RT:TERMS](\sigma[ETERM='IR' \vee ETERM='DB'](\text{THESAURUS}))) \\ &\quad \cup \{'IR', 'DB'\} \\ IRDBRBOOK &= \sigma[INDEX \neq \emptyset](IRDBTERMS \cap /INDEX(BOOKS)) \end{aligned}$$

As another example, assume that we also have the list of references of each book in our database. This information could be represented as an additional relation-valued attribute REFS(RBNO) giving the corresponding book numbers of referenced books. Thus, we have a hypertext-like structure, and we can search for books which are referenced by IR books:

$$\begin{aligned} \text{IRB} &= \sigma['\text{IR}' \in \text{INDEX}](\text{BOOK}) \\ \text{IRREF} &= \sigma[\text{REFS} \neq \emptyset](\Pi[\text{BNO}](\text{IRB})) \cap / \text{REFS}(\text{BOOK}) \end{aligned}$$

Alternatively, we can search for books which reference IR books by

$$\text{IRREFD} = \sigma[\text{RBNO} = \text{BNO}](\Pi[\text{RBNO}](\nu[\text{REFS:RBNO}](\text{IRB})) \times \text{BOOK}).$$

In the two previous examples, the additional information is represented in nested relational form. Now we give two more examples of nested relational structures which are typical for IR applications. First, consider articles appearing in proceedings volume. The structure of proceedings can be represented by means of the following NF2 relation:

PROCEEDINGS (ISBN, PTITLE, EDITOR(NAME), ARTICLE(TITLE, PAGES, AUTHOR(NAME), INDEX(TERM))).

Searching for IR articles in proceedings can be performed by means of the following query:

$$\sigma[\text{ARTICLE} \neq \emptyset](\sigma['\text{IR}' \in \text{INDEX}] / \text{ARTICLE} (\text{PROCEEDINGS}))$$

The structure of journals is more deeply nested:

JOURNAL (ISSN, JTITLE, VOLUME(VNO, YEAR, ISSUE (NUMBER, MONTH, ARTICLE(TITLE, PAGES, AUTHOR(NAME), INDEX(TERM))))))

Here IR articles can be retrieved by

$$\sigma[/\text{VOLUME}/\text{NUMBER}(\text{ARTICLE} \neq \emptyset)](\sigma['\text{IR}' \in \text{INDEX}] / \text{VOLUME}/\text{NUMBER}/\text{ARTICLE} (\text{JOURNAL})).$$

In a typical IR application, a user may not want to formulate specific queries for different types of documents; rather, a single query should be processed on the BOOK, PROCEEDINGS and JOURNAL relations in parallel (as long as the query uses attributes occurring in all three relations). This issue is addressed in object-oriented models, which can be seen as an extension of NF2 models (see [Scholl & Schek 90]). We are currently investigating this direction.

As a major advantage of using DB-oriented data models for IR data, queries are able to retrieve any kind of object stored in the database — not only documents as in IR systems. For example, if we want to know which journals ever published an IR article, we can formulate the query

$$\Pi[\text{ISSN}, \text{JTITLE}](\sigma[/\text{VOLUME}/\text{NUMBER}/\text{ARTICLE}(' \text{IR}' \in \text{INDEX})] (\text{JOURNAL}))$$

Alternatively, we can search for journals publishing IR articles only (that is, all articles are about IR):

$$\Pi[\text{ISSN}, \text{JTITLE}](\sigma[\neg / \text{VOLUME}/\text{NUMBER}/\text{ARTICLE}(' \text{IR}' \notin \text{INDEX})] (\text{JOURNAL}))$$

Since authors also are items in our database, we may want to search for IR authors:

$$\mu[\text{AUTHOR:ANAME}](\Pi[\text{AUTHOR}](\sigma['\text{IR}' \in \text{INDEX}])(\text{BOOK}))$$

Finally, the pNF2 algebra allows for restructuring the data in whatever form the user wants to see it. For example, if we want to have authors together with the list of books they wrote, we can formulate

$$\text{AUBOOK} = \nu[\text{BNO}, \text{YEAR}, \text{INDEX}, \text{PRICE:BOOKS}](\mu[\text{AUTHOR:ANAME}](\text{BOOK})).$$

This gives us a relation with the schema

AUBOOK(ANAME, BOOKS(BNO, YEAR, INDEX(TERM), PRICE(VAL))).

7 Event probabilities

In the examples shown above, we have also given the probabilities for the result tuples. Now we describe how these are computed. In our model, we assume that all event probabilities for tuples in base relations are given explicitly. When we apply pNF2 expressions, then we form event expressions for the tuples of the result relation (e.g. the conjunction of the event keys of the argument tuples in the case of intersection and Cartesian product or the disjunction in the case of union and projection). Based on these event expressions, the event probabilities for tuples in derived relations have to be computed. The crucial point is: Can the event probabilities always be computed?

In general, it is obvious that dependent relations pose problems. However, for most applications, dependent relations will play only a minor role. In most cases with stochastic dependence, it is very difficult to gather the corresponding dependence information; for this reason, it is more feasible to use an independence model as approximation.

In the following, we will focus on databases containing no dependent relations. For this type of databases, we can compute all event probabilities.

Theorem 1 *If a database contains no dependent base relations, then for all derived relations generated by pNF2 expressions, the event probabilities can always be computed.*

The proof of this theorem — which is rather lengthy — is given in [Rölleke 94]. Here we will only describe the algorithm underlying this proof.

In the case of a base relation R , we have to compute for each tuple $r \in \text{val}(R)$ the unconditional probability $P(r.\eta)$, where $r.\eta$ denotes an event expression. Otherwise, if R is a relation-valued attribute of a relation T , then we seek for the conditional probability $P(r.\eta|t.\eta)$ where $r \in \text{val}(R)$, $t \in \text{val}(T)$ and $t.\delta = \text{val}(R)$. With Bayes' theorem, this probability can be transformed to

$$P(r.\eta|t.\eta) = \frac{P(r.\eta \wedge t.\eta)}{P(t.\eta)}.$$

As an example, consider relation IRB, where we get for the IR tuple in INDEX:

$$\beta = P(B\hat{E}I1 \wedge B\hat{E}I1 \wedge B\hat{E}) / P(B\hat{E}I1 \wedge B\hat{E}) = 0.9/0.9 = 1.0$$

In relation BL30A, for BNO=3 the probability for the inner tuple with VAL=28 is computed as:

$$\beta = P(B\hat{E} \wedge (B\hat{E}P4 \vee B\hat{E}P5) \wedge B\hat{E}P4) / P(B\hat{E} \wedge (B\hat{E}P4 \vee B\hat{E}P5)) = 0.7/1.0 = 0.7$$

If R is a subrelation of T , T itself may be a subrelation of another relation U , where $\exists u \in \text{val}(U) t \in u.\delta[T]$. Then — in the same way as in base relations — the tuple t can only exist if $u.\eta = \text{true}$. So we get $P(t.\eta) = P(t.\eta|u.\eta) \cdot P(u.\eta) = P(t.\eta \wedge u.\eta)$. This last step has to be repeated until we reach the top-level relation.

In the general case, assume that relation R is nested at level $l+1$ (top-level relations have level 1), so we have tuples $r, t_l, t_{l-1}, \dots, t_1$ with $r \in t_l.\delta$, $t_l \in t_{l-1}.\delta$, $\dots, t_2 \in t_1.\delta$ where t_1 is a tuple of the top-level relation. Then we can compute the conditional probability $P(r.\eta|t_l.\eta)$ as follows:

$$P(r.\eta|t_l.\eta) = \frac{P(r.\eta \wedge t_l.\eta \wedge t_{l-1}.\eta \wedge \dots \wedge t_2.\eta \wedge t_1.\eta)}{P(t_l.\eta \wedge t_{l-1}.\eta \wedge \dots \wedge t_2.\eta \wedge t_1.\eta)}$$

So we have to compute the ratio of two probabilities of event expressions for tuples in relation-valued attributes, and a single probability of an event expression for tuples of top-level relations.

Now we turn to the problem of computing the probability of event expressions. Since any 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 according to the rules of Boolean algebra into disjunctive normal form (DNF), that is: $\eta = \bigvee_{i=1}^n K_i$, 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). An important feature of event expressions in DNF is that unnegated event keys referring to tuples of relation-valued attributes always occur in conjunction with the event keys of their higher-level tuples (see the examples from the previous sections). Generally speaking, assume that we have an event key of the form $\langle \kappa_1, \kappa_2, \dots, \kappa_{l-1}, \kappa_l, \kappa_R \rangle$, where each κ_i is a pair of relation name and event ID. Then any conjunct containing this event key in unnegated form will also contain the following conjunction

$$\langle \kappa_1, \kappa_2, \dots, \kappa_{l-1}, \kappa_l \rangle \wedge \langle \kappa_1, \kappa_2, \dots, \kappa_{l-1} \rangle \wedge \dots \wedge \langle \kappa_1, \kappa_2 \rangle \wedge \langle \kappa_1 \rangle.$$

This means that we can compute the probability of this conjunct by using the conditional probabilities of our base relations:

$$\begin{aligned}
& P(\langle \kappa_1, \kappa_2, \dots, \kappa_{l-1}, \kappa_l, \kappa_R \rangle \wedge \langle \kappa_1, \kappa_2, \dots, \kappa_{l-1}, \kappa_l \rangle \wedge \langle \kappa_1, \kappa_2, \dots, \kappa_{l-1} \rangle \wedge \dots \wedge \langle \kappa_1, \kappa_2 \rangle \wedge \kappa_1) \\
&= \frac{P(\langle \kappa_1, \kappa_2, \dots, \kappa_{l-1}, \kappa_l, \kappa_R \rangle)}{P(\langle \kappa_1, \kappa_2, \dots, \kappa_{l-1}, \kappa_l \rangle)} \cdot \frac{P(\langle \kappa_1, \kappa_2, \dots, \kappa_{l-1}, \kappa_l \rangle)}{P(\langle \kappa_1, \kappa_2, \dots, \kappa_{l-1} \rangle)} \cdot \dots \cdot \frac{P(\langle \kappa_1, \kappa_2 \rangle)}{P(\kappa_1)} \cdot P(\kappa_1) \\
&= P(\langle \kappa_1, \kappa_2, \dots, \kappa_{l-1}, \kappa_l, \kappa_R \rangle | \langle \kappa_1, \kappa_2, \dots, \kappa_{l-1}, \kappa_l \rangle) \cdot P(\langle \kappa_1, \kappa_2, \dots, \kappa_{l-1}, \kappa_l \rangle | \langle \kappa_1, \kappa_2, \dots, \kappa_{l-1} \rangle) \cdot \dots \\
&\quad \cdot P(\langle \kappa_1, \kappa_2 \rangle | \kappa_1) \cdot P(\kappa_1)
\end{aligned}$$

For negated event keys, we have to modify the DNF in order use our conditional probabilities from the base relations. Negated event keys are produced by set difference and selections involving negated conditions or set comparisons (see e.g. figure 3). Assume that we have a negated conjunct $\neg(\langle \kappa_1, \kappa_2 \rangle \wedge \kappa_1)$. Application of De Morgan's law gives us $\neg\langle \kappa_1, \kappa_2 \rangle \vee \neg\kappa_1$. So $\neg\langle \kappa_1, \kappa_2 \rangle$ may occur in a conjunct of the DNF which does not contain κ_1 as well. However, from the definition of the basic events we know that $\langle \kappa_1, \kappa_2 \rangle$ can only be true if κ_1 is true as well. For this reason, the following equality holds:

$$\neg\langle \kappa_1, \kappa_2 \rangle = \neg\langle \kappa_1, \kappa_2 \rangle \wedge \kappa_1 \vee \neg\kappa_1$$

However, we can show that the negated key $\neg\kappa_1$ is already contained in the DNF (produced by De Morgan's law), so there is no need to add it to the DNF. For the remaining expression, we can use the conditional probabilities given:

$$P(\neg\langle \kappa_1, \kappa_2 \rangle \wedge \kappa_1) = (1 - P(\langle \kappa_1, \kappa_2 \rangle | \kappa_1)) \cdot P(\kappa_1)$$

In a similar way, we can show for the general case of a negated event key that we have to do the following replacement:

$$\begin{aligned}
\neg\langle \kappa_1, \kappa_2, \dots, \kappa_{l-1}, \kappa_l, \kappa_R \rangle &\rightarrow \neg\langle \kappa_1, \kappa_2, \dots, \kappa_{l-1}, \kappa_l, \kappa_R \rangle \wedge \\
&\quad \langle \kappa_1, \kappa_2, \dots, \kappa_{l-1}, \kappa_l \rangle \wedge \\
&\quad \langle \kappa_1, \kappa_2, \dots, \kappa_{l-1} \rangle \wedge \\
&\quad \dots \\
&\quad \langle \kappa_1, \kappa_2 \rangle \wedge \\
&\quad \kappa_1
\end{aligned}$$

Let K'_i denote the conjuncts after this modification. Then we can apply the inclusion-exclusion formula (see e.g. [Billingsley 79, p. 20]):

$$q(\eta) = P\left(\bigvee_{i=1}^n K'_i\right) = \sum_{i=1}^n \left((-1)^{i-1} \sum_{1 \leq j_1 < \dots < j_i \leq n} P(K'_{j_1} \wedge \dots \wedge K'_{j_i}) \right). \quad (1)$$

In order to compute one of the probabilities $p = P(K'_{j_1} \wedge \dots \wedge K'_{j_i})$, the following steps have to be performed:

1. Remove duplicate event keys from the conjunct.
2. If the conjunction contains the same event key in negated and unnegated form then $p := 0$; return.
3. If the conjunction contains different event keys from the same disjoint relation in unnegated form, then $p := 0$; return.
4. For any unnegated event key from a disjoint relation, remove all negated event keys from this relation.
5. For any set of negated event keys from the same disjoint relation, replace this set by a new event key and assign it the counterprobability of the sum of the probabilities of this set of keys (e.g. $P(\bar{a} \wedge \bar{b}) = 1 - (P(a) + P(b))$)
6. $p :=$ the product of the event probabilities of the remaining event keys, using the opposite probability if the event key is negated.

A disadvantage of this procedure is its computational complexity of $O(2^n)$ for a DNF with n conjuncts. However, for many pNF2 expressions it is possible to simplify the computation of event probabilities by directly computing the probability of a result tuple from the probabilities of the event tuple without constructing the event expression. It is the task of the optimizer to decide whether or not an pNF2 expression (or parts thereof) can be processed this way. For this purpose, the optimizer can use knowledge about the type of the probabilistic relations involved and the key attributes of these relations. Of course, when deterministic relations are involved, significant simplifications are possible.

8 Conclusions and outlook

The pNF2 model presented here provides powerful features for modelling imprecise data and for expressing vague queries. So it is an ideal basis for databases with imprecise data, for document retrieval systems and for the integration of IR and database systems.

In this paper, we have not addressed the issue whether the pNF2 model is an algebra or not. In fact, it can be shown that for deterministic relations, the pNF2 model yields the same results as NF2 algebra ([Fuhr & Rölleke 94]). On the other hand, for non-deterministic relations, certain equivalences from NF2 algebra do not hold in general. For example, for relations in partitioned normal form, nest is inverse to unnest in NF2 algebra, but not in the pNF2 model. So there are additional constraints for these equivalences (see [Krüger 95]).

Here we have only described the basic operators of our model which are a result of redefining the operators of an ordinary NF2 algebra. In some cases, this leads to rather complicated expressions when addressing e.g. imprecise attribute values. This is mainly due to the generality of our approach, where we can have arbitrary nesting levels and dependencies between imprecise attribute values. However, given the powerful basic operations, it is no problem to define a less complicated query language for restricted forms of imprecision like in the references cited in the introduction. For certain applications (e.g. statistical databases), additional operators may be necessary, e.g. for conversion between deterministic and disjoint/independent relations and vice versa. An operator of this type would also allow to display the original probabilities of tuples in relation-valued attributes in the output relation. The generalized selection operator described in [Fuhr & Rölleke 96b] also could be included in the pNF2 algebra; this operator is based on the concept of vague predicates, where a selection predicate (e.g. similarity of two attribute values) may have a probabilistic outcome.

In principle, it is also possible to combine our approach with a different theory of uncertainty, like e.g. Dempster-Shafer theory. For this purpose, only the definition of the β part of the tuples and the computation of the uncertainty weights have to be replaced.

Although basing our model on intensional semantics introduces some overhead, we have chosen this approach in order to guarantee results which are consistent with the underlying theory. In addition, there are more equivalences, thus allowing the optimizer to choose among different expressions for computing the result of a query. As mentioned before, the optimizer can also detect expressions for which the event probabilities can be processed more efficiently than with the general algorithm. Currently, we are investigating these equivalences and rules for deciding about the strategy for computing the event probabilities.

We have finished the implementation of a first version of a system based on the pNF2 model recently. Instead of implementing this model as an extension of an NF2 database system, the design of our system is based on the following idea: It is typical for information systems with a high degree of uncertain data (e.g. information retrieval systems) that a user wants to see only the top ranking elements of a query result — in contrast to database systems where the set of answer tuples has to be computed completely. A naive strategy for processing queries would be to compute the result relation first and then to sort it according to decreasing probabilities. However, for many selection conditions it is possible to implement access paths which yield tuples in the order of decreasing probabilities. This ordering can be exploited by a stream-based implementation which reads only as many entries from an access path as are needed for delivering the top ranking output tuples. In [Pfeifer & Fuhr 95], we show that significant efficiency gains can be achieved with this strategy.

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.
- Billingsley, P. (1979). *Probability and Measure*. John Wiley & Sons, Inc, New York.
- Blair, D. C. (1988). An Extended Relational Document Retrieval Model. *Information Processing and Management* 24(3), pages 349–371.
- 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.
- Colby, L. S. (1990). A Recursive Algebra for Nested Relations. *Information Systems* 15(5), pages 567–585.
- 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.; Rölleke, T. (1994). *Towards a probabilistic NF2 algebra*. Technical report, University of Dortmund, Department of Computer Science.
- Fuhr, N.; Rölleke, T. (1996a). *A Probabilistic NF2 Relational Algebra for Imprecision in Databases*. Technical report, University of Dortmund, Department of Computer Science.
- Fuhr, N.; Rölleke, T. (1996b). A Probabilistic Relational Algebra for the Integration of Information Retrieval and Database Systems. (To appear in: *ACM Transactions on Information Systems*).
- IEEE. (1989). *IEEE Data Engineering* 12(2). Special Issue on Imprecision in Databases.
- Jarvelin, J.; Niemi, T. (1995). An NF2 Relational Interface For Document Retrieval, Restructuring and Aggregation. In: Fox, E.; Ingwersen, P.; Fidel, R. (eds.): *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 102–110. ACM, New York.
- Krüger, A. (1995). *Equivalences of probabilistic NF2 algebra expressions*. Diploma thesis, University of Dortmund, Department of Computer Science. (in German).
- Lee, S. (1992). An Extended Relational Database Model For Uncertain And Imprecise Information. In: *Proceedings of the 18th VLDB Conference*, pages 211–220. Morgan Kaufman.
- Macleod, I. (1991). Text Retrieval and the Relational Model. *Journal of the American Society for Information Science* 42(3), pages 155–165.
- Motro, A. (1990). Accommodating Imprecision in Database Systems: Issues and Solutions. *Sigmod record* 19(4), page 69.
- Pfeifer, U.; Fuhr, N. (1995). Efficient Processing of Vague Queries using a Data Stream Approach. In: *Proceedings of the 18th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 189–198. ACM, New York.
- 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.
- Rölleke, T. (1994). *Development of a Probabilistic Relational Data Model*. Diploma thesis, University of Dortmund, Department of Computer Science. (in German).
- Roth, M. A.; Korth, H. F.; Silberschatz, A. (1986). Extended Algebra and Calculus for Nested Relational Databases. *ACM Transactions on Database Systems* 13(4), pages 389–417.
- Sadri, F. (1991). Modeling Uncertainty in Databases. In: *Seventh International Conference on Data Engineering*, pages 122–131. IEEE Computer Society, Los Angeles.
- Salton, G.; Fox, E.; Wu, H. (1983). Extended Boolean Information Retrieval. *Communications of the ACM* 26, pages 1022–1036.
- 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.
- Schek, H.-J.; Scholl, M. (1986). The Relational Model with Relation-Valued Attributes. *Information systems* 2, pages 137–147.
- Scholl, M.; Schek, H.-J. (1990). A Relational Object Model. In: Abiteboul, S.; Kanellakis, P. (eds.): *ICDT '90*, pages 89–105. Springer, Berlin.
- Takahashi, Y. (1993). Fuzzy Database Query Languages and Their Relational Completeness Theorem. *IEEE transactions on knowledge and data engineering* 5(1), page 122.