Probabilistic Reasoning for Large Scale Databases

Thomas Rölleke, Norbert Fuhr
University of Dortmund
[roelleke,fuhr]@ls6.informatik.uni-dortmund.de

Abstract

The complexity of probabilistic reasoning prohibits its application on a large scale of data. In order to reduce the complexity, implementations of modeling approaches restrict themselves with respect to expressive power or relax on the underlying probability theory.

We present the implementation aspects of a probabilistic extension of stratified Datalog. This probabilistic deductive system is strictly based on the well-founded ground of probability theory. The prototypical implementation of the system handles the expensive computation of the probabilities separately from the reasoning process itself. Thus, we can use standard optimization strategies known from deterministic systems in order to cope with large amounts of data.

By adding probabilistic reasoning to a deductive database system we gain the possibility of describing the information retrieval task as computing the probability $P(d \to q)$, i. e. the probability of the inference between a document d and a query q. Therefore, the logical view on databases plus a probabilistic generalization of the data model is a promising candidate for a breakthrough in integrating database and information retrieval technology on the way to multimedia information systems.

1 Introduction

A considerable branch of research done in the field of knowledge representation is driven by the aim to incorporate probability theory in order to reflect the intrinsic uncertainty of knowledge. The injection of probability theory into a data model brings along two major advantages:

1. "The world" with its ubiquitous uncertainty can be modeled more realistically.

2. The result of a query is enriched by probability values which reflect the uncertainty of the inference (reasoning) process.

These advantages are highly desirable in the areas of Artifical Intelligence (AI) and Information Retrieval (IR). But — the high computational complexity of probabilistic reasoning is the price of the richer expressive power when adding probabilities. This complexity often prevents the application of probabilistic data models on a large amount of data.

In this paper we present the implementation aspects of the prototype of a probabilistic extension of Datalog ([Fuhr 95]). The prototype is called HySpirit (Hypermedia System with Probabilistic Inference for the Retrieval of InformaTion) and is aimed at investigating the suitability of uncertain inference and predicate logic for hypermedia information retrieval tasks. We gain a system with the full expressive power of stratified Datalog plus the possibility of probabilistic reasoning. The major features of the system are:

"attached" probability computation: The reasoning process keeps track of the inferences that led to the generation of the IDB (intensional database) facts by maintaining so-called *event expressions*. The operations on event expression correspond to the Boolean operations *and*, *or*, and *not*. When the reasoning process has reached the fixpoint of the tuple computation known from deterministic Datalog, the probability function on the event expressions computes the probability. Thus, the probabilistic reasoning process is two-phased:

1. Computation of the result relation corresponding to the query and alongside construction of the event expressions.
2. Computation of the probabilities by applying a probability function on the event expressions.

This "attached" computation and "localization" of the probability function after the reasoning process for generating the IDB facts allows to investigate and to improve the probability function independently from the reasoning process. The standard optimization strategies known from deterministic deductive systems can be applied in the reasoning process in order to cope with large scale databases.

intensional semantics: For example, in an extensional system the probability $P(A \wedge B)$ is computed as a function of the probability values $P(A)$ and $P(B)$ without considering a possible stochastic dependency between A and B ([Pearl 88]). An intensional system "knows" the probability $P(A \wedge B)$, i. e. it is defined by the semantics.

The semantics of probabilistic Datalog is based on possible worlds and the probability of every combination of events (i. e. every logical formula) is defined as the sum over the probabilities of the worlds where the logical formula holds.

This intensional semantics is assured by using the event expressions for keeping track of the reasoning process. A combination of events, i. e. an instantiation of a rule body, is represented by an event expression and this event expression is assigned to the deduced IDB tuple. The probabilities are computed as a function on event expressions. The probability function is based on the sieve formula in order to "sieve out" the intersection of events (e. g.: $P(A \vee B) = P(A) + P(B) - P(A \wedge B)$). Duplicate events, i. e. events occurring in more than one conjunction, are removed when computing the probability of the intersections. Example: Let A and B be conjunctions, and let E be an event in both conjunctions, and let A' and B' be conjunctions corresponding to A and B but without E, then $P(A \wedge B) = P(A' \wedge E \wedge B' \wedge E) = P(A' \wedge B' \wedge E) \neq P(A) * P(B)$ holds, if $P(E) \neq 1$. An extensional system would compute the probability $P(A \wedge B) = P(A) * P(B)$.

point probabilities: We apply uncertain inference in information retrieval applications — this implies the need to cope with mass data. Thus, the knowledge we derive from the data and store for retrieval purposes is in great parts incomplete in order to cope with the large amount of data at indexing and retrieval time. We argue that only a reasonable restriction of expressive power leads to tractable systems for information retrieval tasks. So we have to find the right tradeoff between expressiveness and computability. A solution for coping with "the large unknown rest of the world" is to choose reasonable assumptions. We make (conditional) independence or disjointness assumptions concerning the probability space. These assumptions lead to point probabilities. We achieve a tractable system and in information retrieval applications we get the desirable ranking of the retrieved objects with a sound meaning according to the probability theory and the underlying assumptions.

2 Background

The issue of probabilistic inference has been studied most extensively in the field of AI. Following more heuristic approaches in the late 70s and the early 80s, the research shifted towards more solid foundation. One line of research focused on probabilistic logics, that is, combining predicate logic with probability theory (see e. g. [Nilsson 86], [Halpern 90], [Fagin & Halpern 94]). However, these theoretical studies are far from being implementable in a system.

Among the more practical approaches, Bayesian inference networks as described in [Pearl 88] are most popular. This approach is restricted to directed acyclic graphs, where the nodes represent probabilistic events (propositions) and the arcs denote probabilistic dependence. Thus, the default assumption is that events are independent, whereas dependence has to be stated explicitly. As another important feature, this approach only deals with point probabilities.

The DUCK system presented in [Güntzer et al. 91] and [Kießling et al. 93] overcomes some of these limitations: It uses probability intervals instead of point probabilities, and independence assumptions must be stated explicitly. The inference engine is based on Datalog with sets and functions. However, probabilistic inference is restricted to propositional logic.

The paper [Schmidt et al. 90] presents an approach for combining Datalog with uncertainty. The work stresses the importance of using optimization strategies like magic sets to increase the efficiency of uncertain reasoning systems. From a probabilistic point of view, this system is based on extensional semantics, and thus the results are consistent with probability theory only in case the inference structure has a tree shape (see [Pearl 88]).

A solid theoretical foundation for probabilistic Datalog is described in [Ng & Subrahmanian 93] and [Ng & Subrahmanian 94]. For each derived fact, a system of linear inequalities is derived during the inference process. Solving this system of inequalities yields the corresponding probability interval. However, no implementation of this approach has been described so far.

In [Poole 93a] and [Poole 93b], a system for probabilistic Datalog is described along with an efficient evaluation algorithm. Here, events are independent by default, and disjointness of events must be stated explicitly. As major restrictions, negation is not supported, and in case there are several rules for the same predicate, the bodies must represent disjoint events. Thus, in terms of relational algebra, this approach neither supports difference nor projection.

Uncertain inference also has a long tradition in information retrieval, starting in the late 50s. In [Rijsbergen 86], it is shown that IR can be interpreted as computing the probability $P(d \rightarrow q)$ that the document d implies the query q. Thus, in comparison with the logical view on databases, both IR and databases are seeking for items implying the query, but IR uses uncertain inference whereas deterministic inference is sufficient in databases.

Since large document collections have to be dealt with in IR, only minimum probabilistic information is applicable. For this reason, cautious approaches to probabilistic reasoning would tend to yield the probability interval [0,1] for most documents retrieved in response to a query, thus being of little use. By assuming the basic events being independent by default, more useful answers can be produced, and numerous evaluations of the effectiveness of these approaches have shown the benefits of this approach. The inference structures typically used in IR are regular and nonrecursive (see the survey on classical IR models in [Wong & Yao 95]). A major step forward was the usage of Bayesian inference networks in the INQUERY system [Callan et al. 92]. However, all these approaches are based on propositional logic — which is sufficient for text retrieval. For multimedia retrieval, however, a more expressive logic is needed, e. g. for modeling spatial or temporal relationships.

The HySpirit System described in this paper is an inference engine for probabilistic stratified Datalog based on intensional semantics. Thus, it overcomes the limitations of the systems mentioned above. HySpirit is aimed at performing efficient probabilistic inference for large data sets, mainly in the area of multimedia retrieval. For this reason, we use simplifying assumptions about the dependence of events: basic events are independent by default, and disjointness of events can be specified explicitly. So the outcomes of the inference process are point probabilities.

By using deductive database technology for information retrieval, HySpirit also offers an ideal basis for the integration of IR and database systems.

3 The Scope of Probabilistic Datalog

Probabilistic Datalog ([Fuhr 95]) is an extension of stratified Datalog [Ullman 88]. The basic idea is to add fact and rule probabilities to a Datalog program (see figure 1). For example, the fact $q(1)$ is true with a probability of 0.9. The rule holds with a probability of 0.8. A rule probability is interpreted as the conditional probability $P(r(1)|q(1)) = 0.8$.

```
0.9 q(1).
0.8 r(X) :- q(X).
```

Figure 1: A probabilistic Datalog program

The semantics of probabilistic Datalog is built on the Herbrand base and possible worlds ([Nilsson 86]). The probability of a fact is equal to the sum over the probabilities of the worlds where the fact is true. That means we assume a set of worlds and we associate with each world w_i a probability $P(w_i)$ and a subset of the Herbrand base which forms the set of true facts in a world (see figure 2).

World	$P(w_i)$	True facts
w_1	0.9 * 0.8	q(1), r(1)
w_2	0.9 * 0.2	q(1)
w_3	0.1 * 0.8	
w_4	0.1 * 0.2	

Figure 2: Possible worlds, probabilities, and true facts (subsets of the Herbrand base)

Several uncertain rules for one predicate must be disjoint according to their rule bodies, i. e. a fact can be deduced atmost by one uncertain rule. Only probabilistic Datalog programs satisfying the disjointness of probabilistic rules can be evaluated without adding further knowledge about the stochastic dependence of rule instantiations. For example, if we would add to the above program a second rule `r(X) :- p(X)` and the fact `p(1)`, then the program would not be evaluable. For computing the probability $P(r(1))$ we would need to know the probability $P(r(1)|q(1) \wedge p(1))$ where $r(1)$ is the deduced fact and $q(1)$ and $p(1)$ are the rule body instantiations. This probability is not derivable given the probabilities $P(r(X)|q(X))$ and $P(r(X)|p(X))$ which correspond to the rule weights.

The set of possible worlds and the corresponding probabilitiy assignment determines also the probability of logical formulas. Probabilistic Datalog allows for specifying two stochastic assumptions: (1) independent tuples (events), (2) disjoint tuples.

The default is the independence assumption on tuples. The example in figure 2 shows that the probability of the fact $q(1)$ is equal to the sum $P(w_1) + P(w_2)$, i. e. the sum over the probabilities of the worlds, where the fact is true. For computing the probability of a conjunction of independent events, we use the product of the probabilities. For example, consider the two facts $q(1)$ and $q(2)$, the probability $P(q(1)\&q(2))$ is then equal to $P(q(1)) * P(q(2))$. The probability of the conjunction is equal to the sum over the probabilities of the worlds, where both facts are true.

The second assumption — the disjointness assumption — can be declared explicitly by defining a *disjointness key* for a relation. Tuples having the same disjointness key value are treated as disjoint events. For illustrating the usage of the disjointness assumption, consider the relation book in figure 3 with three attributes and a probability value P for each tuple. The attributes price and year are uncertain and this uncertainty can be handled by a discrete probability function (see [Barbara et al. 92]). This means that the probabilities of the tuples with the same bookid but different attribute values for the uncertain attributes sum up to 1 — these tuples represent disjoint events. The probability of a conjunction like book(b1,16,96) & book(b1,26,96) is then equal to zero.

book			
P	bookid	price	year
0.5	b1	16	96
0.4	b1	26	96
0.1	b1	15	95
0.8	b2	10	90
0.2	b2	15	95

Figure 3: The relation book

We use the clause `#book(dk,av,av)` to declare that all tuples of the relation book which correspond in the first attribute value (dk = disjointness key value, av = attribute value) should be treated as disjoint by the probability function. In the independence case, we would have `#book(dk,dk,dk)`, i. e. no tuple corresponds in the disjointness key value with any other tuple. Disjointness means for a subset of the Herbrand base associated with a world that at most one fact of a set of disjoint facts occurs in the set of true facts — thus the disjointness key is the key of a relation in a possible world. Tuples differing in the disjointness key value are independent, i. e. two tuples of relation book differing in their bookid are independent and the corresponding facts may be true in the same world, whereas at most one fact with bookid b1 is true in one world.

The disjointness key also supports the possibility for specifying a probability distribution on dependent events like the attribute values of price and year. The probability of an event is defined semantically as the sum over the world probabilities. In case of a disjointness key the disjoint facts occur in different worlds. The probability of a fact like book-year(b1,96) coming from a projection is then defined as the sum over all disjoint worlds (tuples), where the bookid and the year have the corresponding values — we get for book-year(b1,96) the probability $0.5 + 0.4 = 0.9$. Thus, we can use the disjointness key for specifying for dependent events the probablities of their conjunctions and from this we can also compute the probabilities of the single events.

The next example in figure 4 indicates the application of probabilistic Datalog to IR and demonstrates the difference between intensional and extensional semantics. The predicate (relation) term models the occurrence of terms in documents. The predicate link reflects the link structure among documents. The relation about contains all tuples of the relation term and a document D is also about a term T if there is a link from document D to a document $D1$ which is about term T.

```
0.9 term(ir,d1).
0.8 term(db,d1).
0.5 link(d2,d1).
about(T,D) :- term(T,D).
about(T,D) :- link(D,D1) & about(T,D1).
```

Figure 4: Information Retrieval with exploitation of the link structure

This example points out the motivation for defining probabilistic Datalog: we gain a powerful data model based on probabilistic logic which allows to use the expressive power of predicate logic in information retrieval applications and which supports the modeling of various retrieval functions.

A major feature of probabilistic Datalog is its *intensional semantics*. When querying for all documents about ir and db (?- about(ir,D) & about(db,D)) the reasoning process deduces $\{$0.72 d1, 0.36 d2$\}$. For both documents the corresponding weight gives the probability that the document is about both terms. An *extensional system* — which is characterized by a direct computation of a new probability value as a function of the probabilities of the current inference step — would compute $0.5 * 0.8 * 0.5 * 0.9 = 0.18$ for d2, i. e. the fact link(d2,d1) would be considered twice. From a stochastical point of view an extensional system may therefore yield "wrong" results, whereas the intensional systems compute the stochastically "correct" results.

The intensional semantics in probabilistic Datalog is assured, since the evaluation method is based on the probabilistic relational algebra (see [Fuhr & Rölleke 96]). The basic idea is to introduce so-called *event expressions* in order to encounter stochastic dependencies which may occur when evaluating relational algebra expressions. In principle, each basic tuple (fact) is identified by an atomic event expression and each deduced tuple is accompanied by an event expression reflecting the history of the tuple's generation. While evaluating the Datalog equations only the tuples with the corresponding event expressions are generated. Afterwards, the probability function is applied on the event expressions to compute the resulting probability values.

4 HySpirit — A Prototypical Implementation of Probabilistic Datalog

The evaluation of a probabilistic Datalog program is two-phased:

1. The probabilistic result relation corresponding to the query is computed. Along with the relational algebra operations the event expressions are constructed (see [Fuhr & Rölleke 96]) for reflecting the proofs (rule instantiations) that led to the generation of the tuples.

2. The probabilities of the tuples are computed as a function on the corresponding event expressions.

The major issue of our implementation is that in the first step we can use the same evaluation and optimization strategies as developed for deterministic Datalog. The additional overhead for generating event expressions does not increase the complexity of evaluating the Datalog equations. The higher complexity of probabilistic reasoning comes with the attached computation of the probabilities.

4.1 Computing Probabilistic Relations

The evaluation of a probabilistic Datalog program is structured into three main steps:

1. Given a query, the magic set rewriting method generates a modified program in order to take advantage of early bindings of variables. This results in early selection operations on relations and avoids the computation of the unnecessary relations.

2. The magic set modified program is translated into relational algebra expressions. Since the equivalences of the relational algebra do hold for the probabilistic relational algebra, we can apply the corresponding rules of algebraic optimization (for example: move selections to inner expressions).

3. The Datalog equations are evaluated for each stratum using the semi-naive evalutation algorithm ([Ullman 88]). The evaluation computes iteratively the relations corresponding to the predicates defined by rules. It is an iteration over a set of Datalog equations in order to evaluate the recursive algebra expressions. The evaluation stops if the set of computed tuples is a fixpoint, i. e. the set of computed tuples in iteration step $i-1$ is equal to the set of tuples in iteration step i.

These three steps sketch the computation of the result relation according to a posed query. Apart from the maintenance of event expressions, it is exactly the same as known from non-probabilistic Datalog. The event expressions are built when generating the tuples of the IDB relations. This is done by using the Boolean connectors *and*, *or*, and *not* for combining the event expressions of tuples according to the relational algebra operations selection, projection, union, difference, join (see [Fuhr & Rölleke 96]). When the evaluation has reached the fixpoint of the computation of the probabilistic relations, the iteration over the Datalog equations ends and the probabilities of the tuples are computed as a function on the event expressions.

In the following, we illustrate the construction of event expressions in more detail along with examples. Consider the program shown in figure 5 for computing the transitive closure of the relation link.

```
0.9 link(a,b).
0.8 link(a,c).
0.7 link(a,d).
0.6 link(b,c).
0.5 link(b,d).
0.4 link(c,d).
path(X,Y) :- link(X,Y).
path(X,Y) :- link(X,Z) & path(Z,Y).
```

Figure 5: Relation path is the transitive closure of relation link

The graph represented by the link relation is acyclic and the link relation already contains every possible path. Thus, the first iteration step on the Datalog equations already computes the complete path relation. Figure 6 shows the probabilistic relation path$^{(1)}$ after the first iteration. The first column (η) contains the event expression. Given a fact, η yields the corresponding event expression. In the first iteration, only the first rule adds tuples to path, since path itself (path$^{(0)}$) is empty. The event expressions represent the event (the rule body instantiation) that led to the generated tuples.

In the second iteration step on the set of Datalog equations, the relation path$^{(2)}$ is computed. Now both rules add tuples to the path relation and path$^{(2)}$ is computed as the union of the set

Relation $\text{path}^{(1)}$		
η	X	Y
η(link(a,b))	a	b
η(link(a,c))	a	c
η(link(a,d))	a	d
η(link(b,c))	b	c
η(link(b,d))	b	d
η(link(c,d))	c	d

Figure 6: The probabilistic relation path after the first iteration step

of tuples produced by each rule. Figure 7 shows the probabilistic relation. As an example for a construction of an event expression, consider the fact path(b,d). The event expression η(path(b,d)) is equal to η(link(b,d)) $\vee$ η(link(b,c)) $\wedge$ η(path(c,d)). The first conjunction of the disjunction comes from the first rule, the second from the second rule. The disjunction reflects the union of the relations corresponding to each rule, the conjunction reflects the join operation in the second rule. The function η yields the event expression of a fact. The event expression of an EDB (extensional database) relation (like link) is constant, thus we may omit the function symbol η for EDB events. For IDB facts the event expression is not constant: it may alter in the next iteration.

Relation $\text{path}^{(2)}$		
η	X	Y
link(a,b)	a	b
link(a,c) $\vee$ link(a,b) $\wedge$ η(path(b,c))	a	c
link(a,d) $\vee$ link(a,b) $\wedge$ η(path(b,d)) $\vee$ link(a,c) $\wedge$ η(path(c,d))	a	d
link(b,c)	b	c
link(b,d) $\vee$ link(b,c) $\wedge$ η(path(c,d))	b	d
link(c,d)	c	d

Figure 7: Relation path after the second iteration step

The example makes evident that event expressions of IDB facts may still alter in the last iteration of the Datalog evaluation. But after this last iteration, we have reached also a fixpoint for the event expressions which means that the reasoning process may halt here although it is non-deterministic. Why is it a fixpoint concerning the event expressions? Assume we would compute another iteration step $i + 1$. Each rule body would be instantiated in the same way as in step i, this means we would generate the same conjunctions of event expressions as in step i which leads to the same event expression as in step i. It follows, that the event expression of a fact x in step i is complete, since we use the function η to refer to the facts which were used to deduce the fact x. Thus, the event expression of a fact x contains the whole proof of every fact that was used for deriving fact x. Formally:

Theorem 1 *If the evaluation process of a probabilistic Datalog program has reached a tuple fixpoint (in the deterministic sense), then the construction of the event expressions has also reached its fixpoint.*

Proof 1 *Let T be the set of facts (the computed model) in iteration step $i-1$. The iteration step i leaves T unchanged, which means that a (tuple) fixpoint is reached.*

This implies that at time of evaluating the Datalog equations in step i, each base relation (IDB and EDB) occurring in the relational algebra expression is complete, i. e. each base relation contains all its tuples. The (non-expanded) event expressions of the tuples of the result relation depend on the state (set of tuples) of the base relations occurring on the right side of the Datalog equation. Since this state would not change in step $i+1$, the same event expressions would be constructed in further iteration steps. Thus, the evaluation process has also reached a fixpoint for event expressions.

We have shown that the fixpoint of evaluating the Datalog equations is also a fixpoint of event expressions. This implies that our evaluation process for a probabilistic Datalog program halts due to the same criteria as for the classical (non-probabilistic) Datalog programs.

4.2 Expanding Event Expressions and Computing Probabilities

Before applying the probability function, the event expressions are expanded. Consider again the event expression of the fact path(b,d). In η(path(b,d)) occurs the reference η(path(c,d)). The application of η yields link(c,d) and thus the whole expansion yields the expanded event expression link(b,d) $\vee$ link(b,c) $\wedge$ link(c,d).

Now we can apply the sieve formula. Let $K_1 \vee ... \vee K_n$ be a disjunctive event expression where each K_i is a conjunction of negated or non-negated atomic event expressions.

$$P(K_1 \vee ... \vee K_n) = \sum_{i=1}^{n}(-1)^{i-1}\left(\sum_{1\leq j_1<...<j_i\leq n} P(K_{j_1} \wedge \ldots \wedge K_{j_i})\right)$$

This probability function on event expressions assures that the probability of basic events which occur in more than one conjunction of the event expression is not treated extensionally. Furtheron, it encounters the disjoint events. Thus, the probability computation obeys the intensional semantics.

When evaluating the function, we get in the worst case a complexity of $O(2^n)$, where n is the amount of conjunctions in the event expression and 2^n is the amount of elements in the sum of the sieve formula.

4.2.1 Recursive Event Expressions

In the following, we demonstrate the expansion of recursive event expressions. The basic idea is that an alternative way for proving a fact x which contains the proof itself does not affect the probability of the proof. Consider the example shown in figure 8. It represents a cyclic link structure, since we include the two facts link(a,b) and link(b,a).

Relation path$^{(1)}$		
η	X	Y
link(a,b)	a	b
link(b,c)	b	c
link(b,a)	b	a

Figure 8: A cyclic link structure

Relation path$^{(2)}$		
η	X	Y
link(a,b)	a	b
link(a,b) $\wedge$ η(path(b,c))	a	c
link(b,c) $\vee$ link(b,a) $\wedge$ η(path(a,c))	b	c
link(b,a)	b	a

Figure 9: A cyclic link structure

The cyclic structure in the EDB facts leads to recursive event expressions shown in figure 9: the event expression of path(a,c) contains the event expression of path(b,c) and vice versa.

The probabilistic interpretation of the expansion of recursive event expressions is as follows: If an event expression contains a pointer to itself, the conjunction where it occurs represents a subset of the elementary events of the probability space that form the whole event expression. Thus, this subset is included in the non-recursive part of the event expression.

Theorem 2 *An event expression containing a conjunction with a recursive reference is equivalent to the event expression without that conjunction.*

Proof 2 *Let x be a fact with the event expression $\eta(x) = A \vee (B \wedge \eta(x))$. A is the non-recursive part of the disjunction, and in the conjunction $B \wedge \eta(x)$ the event expression $\eta(x)$ is referred recursively. The event expression contains a non-recursive part, since atleast one rule body instantiation must have been true independently from the fact x itself. Otherwise, the fact would not have been deduced.*

The expansion of the recursive event expression leads to a chain of conjunctions, which can be reduced to $P(A)$ according to the inclusion property $P(A \vee (A \wedge B)) = P(A)$. The inclusion property follows directly by applying the sieve formula:

$$\begin{aligned} P(A \vee (A \wedge B)) &= P(A) + P(A \wedge B) - P(A \wedge B) \\ &= P(A) \end{aligned}$$

Thus we get for the expansion of the event expression:

$$\begin{aligned} P(\eta(x)) &= P(A \vee (B \wedge \eta(x))) \\ &= P(A \vee (B \wedge A) \vee (B \wedge B \wedge \eta(x))) \\ &= P(A \vee (B \wedge A) \vee (B \wedge \eta(x))) \\ &= P(A) \end{aligned}$$

We have shown that the expansion process can cope with recursive event expressions by dropping the conjunction containing the recursive reference and therefore a cyclic structure of EDB tuples is allowed for probabilistic Datalog programs.

We achieve a probabilistic reasoning system, where the number of iterations on the Datalog equations to compute the relational fixpoint is the same as in non-probabilistic Datalog. The relational fixpoint is also the fixpoint of event expressions. Attached to the relational computation, the expansion of event expressions and the computation of the probabilities using the sieve formula is processed.

5 Performance

In our benchmarking testbed, we investigate the variation of two parameters:

1. The size of the document corpus.
2. The complexity of the event expressions.

The size of the document corpus is reflected by a deterministic relation doc. The complexity of the event expressions is determined by the structure of the relation link as already used above.

We pose the query "?- doc(a,D,0) & path(0,N)" to the HySpirit system. The first parameter of relation doc (a) serves to select a number of retrieved documents. It ranges from 'a' to 'z'. Thus, the query retrieves 1/26 of the whole document corpus. The second parameter (D) serves to take the document identifiers of the retrieved documents. The third parameter (0) connects the document with the event expression of the path from node 0 to node N. For the amount of retrieved documents, the HySpirit system has to evaluate the corresponding event expressions. The benchmark run evaluates the same event expression for every retrieved document, but since the sieve formula is applied independently for each tuple, we can use this strategy for simulating typical IR applications.

Figure 10 shows the result of several runs executed by HySpirit. The solid line depicts the performance of the run with a non-probabilistic link structure "0 - 11 - 2", i. e. a link from the source node 0 via a node 11 to the sink node 2. In order to describe link structures, we refer to this structure as a 1-2-run, which means that it has breadth 1 and depth 2. The second parameter of the path predicate (N) in the query is set to the number of the sink (the depth). The dashed lines reflect the probabilistic runs — the shorter the dashes, the higher is the complexity of the event expressions.

The longest dashed line is executed with a link structure of breadth 8 and depth 2 — an 8-2-run. This run leads to the event expression link(0,11) $\wedge$ link(11,2) $\vee$ link(0,21) $\wedge$ link(21,2) $\vee$ ... $\vee$ link(0,81) $\wedge$ link(81,2). The event expression contains 8 conjunctions (determined by the breadth of the link structure) and each conjunction contains 2 link events (determined by the depth of the structure).

The 8-2-run and the 8-4-run nearly coincide with the deterministic run. Thus, HySpirit shows for up to 8 disjunctions with 4 elements per conjunction only a minor difference in computational behaviour comparing to a pure deterministic evaluation. This corresponds to retrieving a document according to eight rule instantiations, and each rule body contains four subgoals.

The shorter dashed lines show the computational behaviour for a 10-2-run and a 10-4 run. Here, the exponential complexity of the sieve formula is evident. The computational behaviour can be improved by exploiting the structure of the event expression (disjointness of events, inclusion of conjunctions) or by computing the counter probability $P(A \vee B) = 1 - P(\neg(A \vee B)) = 1 - P(\neg A \wedge \neg B)$. This computation is cheaper if the number of elements in the biggest conjunction is less than the number of conjunction in the original disjunctive normal form.

As a second aspect we want to show that the size of the conjunction does not affect the computational behaviour of a probabilistic evaluation if the event expression contains exactly one conjunction.

The run is executed with a cycle, i. e. link(1,2), link(2,3), ..., link(n-1,n), link(n,1), and with the query "?- path(1,X)" to compute all paths starting from node 1. Since there is exactly one path from node 1 to each other node, each event expression contains exactly one conjunction. For example, we get η(path(1,3)) = link(1,2) $\wedge$ η(path(2,3)). The conjunction η(path(1,1)) is the longest one and contains n link events.

Figure 11 shows that the two curves for the deterministic and probabilistic evaluation are almost the same. The additional overhead for maintaining the conjunctions and computing the

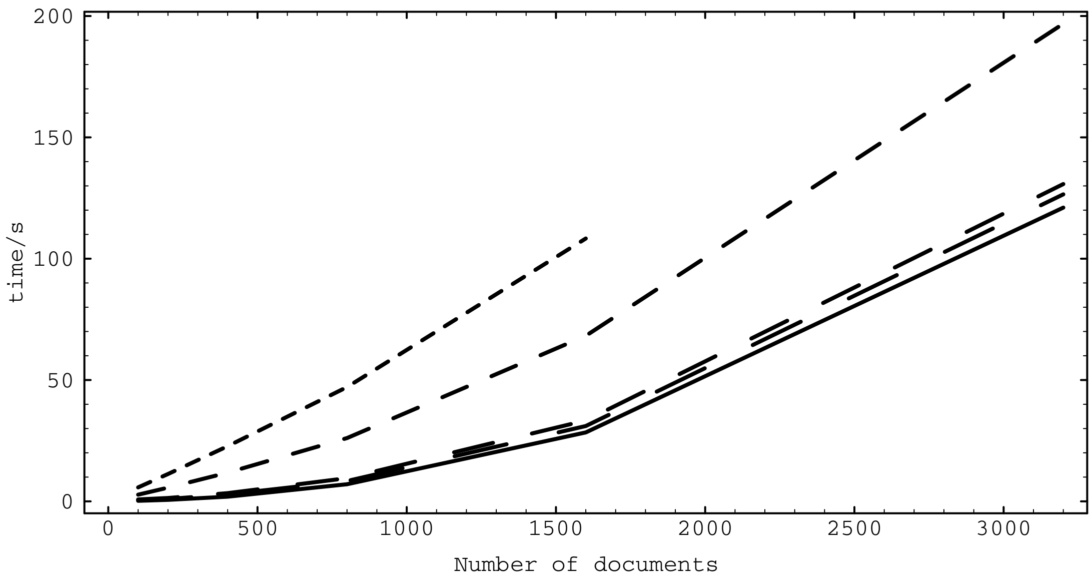

Figure 10: Five benchmarks: solid line = deterministic run, dashed lines = four probabilistic runs with increasing complexity of the event expressions

probabilities does not increase the complexity of the reasoning process if the event expression contains only one conjunction.

The complexity has a quadratic part which comes from our current join and union implementation. We will improve this behaviour by implementing these operations more efficiently (index over the tuples of a relation and merge join and union implementations).

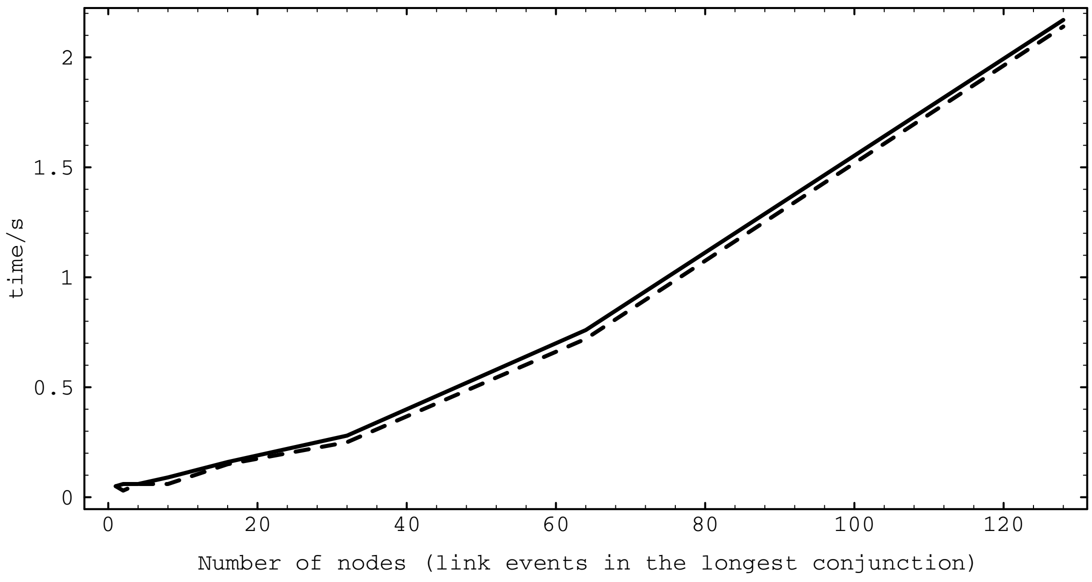

Figure 11: Two benchmarks: solid line = deterministic run, dashed line = probabilistic run

We have used the new programming language BETA for implementing HySpirit. The major feature of this object-oriented language is the unified view (patterns) on data and functions which increases the reusability and supports the software modularity. The BETA compiler currently does little optimization, thus the absolute performance of HySpirit can be expected to improve with future versions of the BETA system.

6 Conclusions and Outlook

Probabilistic reasoning is a desirable aim for being able to model "the world" more realistically. And — the result of a query bears a sorting order according to the uncertainty of the inference. Both, the modeling aspect and the order of the result are of high importance in IR applications.

Typically, IR applications deal with large sets of data which often prohibits the use of probabilistic reasoning. The HySpirit system enables us to apply uncertain inference to large data sets. Thus, we get a system which supports the investigation of new directions in information retrieval, i. e. the usage of predicate logic in multimedia retrieval and the description of the retrieval process as an uncertain inference process.

With regard to the operations on the tuples (relations), we gain an implementation with the complexity and optimization possibilities of deterministic Datalog. The attached probability computation has an exponential complexity according to the number of conjunctions in a disjunctive event expression.

The localization of the probability function bears further optimization possibilities. For example, we can do without event expressions if the extensional result is equivalent to the intensional one. And the application of the sieve formula becomes cheaper if we exploit the underlying structure of the event expressions (disjoint events, inclusion of conjunctions). It is also possible to consider the sieve formula evaluation as an anytime algorithm, i. e. the computation is stopped after a certain number of elements of the sum. The more elements are included, the smaller becomes the probability interval and we can define an approximation of the probability value.

Furtheron, we are currently implementing a connection with an external database (tuple container) for maintaining the EDB relations. This strategy saves the parsing process for loading the EDB database. Only those EDB tuples selected from the external database by the selection operations generated by the magic rewriting process must be loaded to main memory in order to process the reasoning process. Thus, we can keep the large EDB database outside our engine and we can use the administration tools of an external database management system in order to maintain the large amounts of data we have to deal with in multimedia information retrieval applications.

In [Rölleke & Fuhr 96] we have presented a model for retrieving multimedia objects. This model considers inconsistencies and uses predicate dependent closed-world and open-world assumptions. We map this model onto a four-valued logic which allows for coping with inconsistencies and both assumptions for handling negation. This four-valued logic can be expressed via probabilistic Datalog programs and thus, we gain the possibility to use the HySpirit system for demonstrating multimedia information retrieval using probabilistic inference.

Acknowledgement
This work is supported in part by the ESPRIT BRA project 8134 (FERMI). We would like to thank Christian Altenschmidt, Claus-Peter Knaut, and Achim Oberreuter for their enthusiastic work on the HySpirit prototype and their comments and suggestions.

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.

Callan, J.; Croft, W.; Harding, S. (1992). The INQUERY Retrieval System. In: *Proceedings of the 3rd International Conference on Database and Expert Systems Applications,* pages 78–83.

Fagin, R.; Halpern, J. (1994). Reasoning About Knowledge and Probability. *Journal of the ACM 41(2),* pages 340–367.

Fuhr, N.; Rölleke, T. (1996). A Probabilistic Relational Algebra for the Integration of Information Retrieval and Database Systems. (To appear in: ACM Transactions on Information Systems).

Fuhr, N. (1995). Probabilistic Datalog - a Logic for Powerful Retrieval Methods. 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 282–290. ACM, New York.

Güntzer, U.; Kießling, W.; Thöne, H. (1991). New Directions for Uncertainty Reasoning in Deductive Databases. In: Clifford, J.; King, R. (eds.): *Proceedings of the ACM SIGMOD International Conference on the Management of Data*, pages 178–187. ACM, New York.

Halpern, J. Y. (1990). An Analysis of First-Order Logics of Probability. *Artificial Intelligence 46,* pages 311–350.

Kießling, W.; Köstler, G.; Güntzer, U. (1993). Fixpoint Evaluation with Subsumption for Probabilistic Uncertainty. In: Stucky, W.; Oberweis, A. (eds.): *Datenbanksysteme in Büro, Technik und Wissenschaft,* pages 316–333. Springer, Berlin et al.

Ng, R.; Subrahmanian, V. S. (1993). A Semantical Framework for Supporting Subjective and Conditional Probabilities in Deductive Databases. *Journal of Automated Reasoning 10,* pages 191–235.

Ng, R.; Subrahmanian, V. S. (1994). Stable Semantics for Probabilistic Deductive Databases. *Information and Computation 110,* pages 42–83.

Nilsson, N. J. (1986). Probabilistic Logic. *Artificial Intelligence 28,* pages 71–87.

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

Poole, D. (1993a). Logic Programming, Abduction and Probability. *New Generation Computing 11(3),* pages 377–400.

Poole, D. (1993b). Probabilistic Horn abduction and Bayesiam networks. *Artificial Intelligence 64,* pages 81–129.

van Rijsbergen, C. J. (1986). A Non-Classical Logic for Information Retrieval. *The Computer Journal 29(6),* pages 481–485.

Rölleke, T.; Fuhr, N. (1996). Retrieval of Complex Objects Using a Four-Valued Logic. In: *Proceedings SIGIR'96*. ACM, New York.

Schmidt, H.; Steger, N.; Güntzer, U.; Kiessling, W.; Azone, R.; Bayer, R. (1990). Combining Deduction by Certainty with the Power of Magic. In: Kim, W.; Nicolas, J.; Nishio, S. (eds.): *Deductive and Object-Oriented Databases,* pages 103–122. Elsevier Science Publishers, North-Holland.

Ullman, J. (1988). *Principles of Database and Knowledge-Base Systems,* volume I. Computer Science Press, Rockville (Md.).

Wong, S.; Yao, Y. (1995). On Modeling Information Retrieval with Probabilistic Inference. *ACM Transactions on Information Systems 13(1),* pages 38–68.