

Probabilistic Ontologies and Relational Databases

Octavian Udrea¹, Yu Deng¹, Edward Hung², and V.S. Subrahmanian¹

¹ University of Maryland, College Park MD 20742, USA
{udrea,yuzi,vs}@cs.umd.edu

² The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong**
csehung@comp.polyu.edu.hk

Abstract. The relational algebra and calculus do not take the semantics of terms into account when answering queries. As a consequence, not all tuples that should be returned in response to a query are always returned, leading to low recall. In this paper, we propose the novel notion of a *constrained probabilistic ontology* (CPO). We developed the concept of a CPO-enhanced relation in which each attribute of a relation has an associated CPO. These CPOs describe relationships between terms occurring in the domain of that attribute. We show that the relational algebra can be extended to handle CPO-enhanced relations. This allows queries to yield sets of tuples, each of which has a probability of being correct.

1 INTRODUCTION

Over the years, there has been an implicit assumption that relational databases correctly answer queries. However, relational databases are being increasingly populated by people who know very little about how databases work. They are also being used more and more by people who do not know much about the innards of a DBMS. It is therefore important that the DBMS must be smart enough to return answers that have *both* a high precision and a high recall. Databases today do very well in terms of precision, but not as well with respect to recall. Our goal in this paper is to *compute* probabilistic answers to queries internally, but only return a tuple to the user if its probability of being correct exceeds a given threshold p_{thr} . As a consequence, the end user does not even see the probabilities (either in the data or in the answer) unless he really wants to. For example, suppose the user poses a query Q . Our PARQ (Probabilistic Answers to Relational Queries) system may compute that tuple t should be in the answer with 90% probability. If the threshold $p_{thr} \leq 0.9$, then t will be returned to the user, otherwise not. Thus, PARQ's use of probabilities is completely hidden from the user - the inputs are the same old relations and queries, and the output is also just an ordinary relation. The probabilities are

** Work performed while at the University of Maryland, College Park MD 20742, USA.

only used internally by the system. We should emphasize, however, that a DB administrator can and should have visibility into some of the workings of PARQ.

This raises two questions: (I) How should we assign a probability to a tuple t being in the answer to query Q , and (II) how should the threshold p_{thr} be set? Most of this paper will focus on answering question (I). Question (II) will be answered experimentally towards the end of this paper.

The organization and contributions of this paper are summarized below.

1. We first provide a detailed motivating example from the domain of astronomy in Section 2.
2. Our first major contribution is the concept of a *constrained probabilistic ontology* (CPO) introduced in Section 3. We introduce the notion of a *CPO-enhanced relation* in which a CPO is associated with each attribute of a relation.
3. In Section 4, we provide a formal definition of what it means for a CPO to be consistent. We show that checking consistency of CPOs is NP-complete. Fortunately, there is a class of CPOs that are guaranteed to be consistent and there are polynomial algorithms to check if a given CPO belongs to this class or not.
4. When performing various operations, we might need to merge two CPOs together. For example, when doing a condition-join between two CPO-enhanced relations with the condition $A_1 = A_2$, we might need to merge the CPOs associated with A_1 and A_2 . We develop the **CPOMerge** algorithm that merges two CPOs together in the presence of certain “interoperation constraints” linking together terms in the two ontologies in Section 5.
5. In Section 6, we extend the relational algebra to utilize CPOs. In particular, we define selection, join, and the set operations. Projection is unaffected by CPOs.

2 MOTIVATING EXAMPLE

There are numerous astronomical databases³ in existence today. These, in turn, are accompanied by several *stellar classification schemes* based on diverse characteristics such as luminosity, mass, size, composition, etc. For instance, the most popular scientific classifiers are the *spectral class classification* and the *Yerkes luminosity classes*. Uncertainty in star classifications occurs for many reasons: first, some celestial bodies are discovered purely by mathematical means and most classifiers are based on direct observation of the body in question. Secondly, even with the possibility of direct observation there is often not enough information to assume that a star belongs to some specialized class. One may also need to classify a star according to multiple scientific classifiers at once. Figure 1 (without the underlined elements) shows a simple CPO (we will give the formal details later) for holding star data, together with an ontology for star

³ A quick Google search on *astronomy database* reveals over a million hits - the first five pages returned led to several databases.

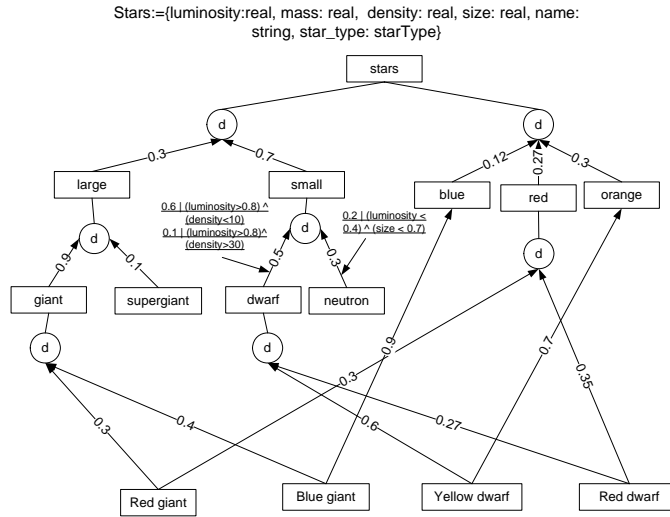


Fig. 1. A simple CPO.

classification. The stars are classified according to size, brightness and color. Each such *decomposition* is *disjoint* and mutually disjoint classes are joined by a "d". For instance, stars that are **red** cannot be **yellow** at the same time. We can easily see that the ontology in this example is an **is-a** graph with arrows indicating the relationship. The labels on the edges of the graph have an intuitive meaning: if edge $u \rightarrow v$ is labeled with quantity p then there is a p probability that an arbitrary object in class v is in fact in class u . The underlined elements in Figure 1 represent *constrained probabilities*, which give circumstances in which the default probability of an edge may change. Using such an ontology, users may ask queries such as:

- **Select all stars that are small.** Clearly, any entity listed explicitly as a small star should be returned. However, in addition, any entity listed as a **dwarf**, **neutron**, etc. should be returned using class-subclass relationships. Now consider an entity e listed in a database as a *star*. There is a 70% probability that any star is small - so if $p_{thr} \leq 0.7$, e should be returned as a response to this query.
- **Join example** Given two relations containing star data with attributes $\mathcal{R}_1 = \{name, density, star_type\}$ and $\mathcal{R}_2 = \{mass, location, star_type\}$, we want to join these relations based on the *star_type* attribute. Classical relational join cannot capture the fact that different values in the *star_type* attribute can in fact refer to the same set of objects and thus will not return all tuples. If we know for instance that a **red** star is the same as a **crimson** star, using CPO enhanced relations we can also join the tuples that have $\mathcal{R}_1.star_type = red$ and $\mathcal{R}_2.star_type = crimson$.

- **Select all stars that are dwarf with a 50% probability or more.** This kind of query cannot be answered using a relation alone (with no probabilities encoded in it). For a simple condition such as *star_type* = **dwarf** it adds a *certainty threshold*. The result of such an operation would consist of all stars that have *star_type* either **dwarf**, **yellow_dwarf**, **red_dwarf** as well as all stars that have *star_type*=**small**, since there is a 0.5 probability that these are **dwarf** as well. We define such a selection operation in Section 6.

We will use these examples throughout the rest of this article to illustrate our definitions. By the end of the article we will have defined selection, projection, join and set operations for CPO-enhanced relations.

3 CONSTRAINED PROBABILISTIC ONTOLOGY (CPO)

Eiter et. al. [1] proposed the concept of a *probabilistic object base* (POB) in which they defined a structure called a POB-schema. They then showed how to embed probabilities into object bases. In contrast, we will slightly modify their POB-schema definition and then use it to define our main structure, a CPO. Of course, we use CPOs in a very different way than Eiter et. al.[1] — we use it to improve the recall of answers to queries posed to relational databases with no probabilities anywhere in the DB.

Definition 1. (*POB-schema*). A *POB-schema* is a quadruple $(\mathcal{C}, \Rightarrow, \text{me}, \psi)$ where:

- (1) \mathcal{C} is a finite set of classes.
- (2) \Rightarrow is a binary relation on \mathcal{C} such that $(\mathcal{C}, \Rightarrow)$ is a directed acyclic graph. The relation $c_1 \Rightarrow c_2$ says that the class c_1 is an immediate subclass of c_2 .
- (3) *me* maps each class c to a set of disjoint subsets of the immediate subclasses of c . In our example, $\text{me}(\text{stars}) = \{\{\text{large}, \text{small}\}, \{\text{blue}, \text{red}, \text{orange}\}\}$. *me* produces clusters corresponding to the disjoint decompositions of a class.
- (4) ψ maps each edge in $(\mathcal{C}, \Rightarrow)$ to a positive rational number in the interval $[0,1]$ such that for all classes c and all clusters $\mathcal{L} \in \text{me}(c)$, it is the case that $\sum_{d \in \mathcal{L}} \psi(d, c) \leq 1$. This property will ensure that the sum of the probabilities on the edges originating from a disjoint decomposition is less than or equal to 1.

If we strip away all the conditional probabilities in Figure 1 (i.e. the condition probabilities from the node **small** to **dwarf** and **neutron**, then we would have a POB-schema. Our definition is identical to that in [1] except for the tuple types associated with the classes in the graph. We now define the transitive closure of the \Rightarrow relation in a POB-schema.

Definition 2. (\Rightarrow^*). The \Rightarrow^* relation is the reflexive, transitive closure of \Rightarrow .

A CPO is obtained from a POB-schema by allowing edges to be labeled with multiple *constrained/conditional probabilities*. When associating a relation with

a POB-schema, one may want to express that when certain conditions are met, the probability of an object being in a certain subclass of a given class increases or decreases. For instance, in Figure 1, we might want to express the fact that when *luminosity* > 0.8 (which is true of our sun), the probability that a **small** star is a **dwarf** increases to 0.75.

Definition 3. (*Simple constraint*). A simple constraint is of the form $\mathcal{A}_i \in \mathcal{D}$, where \mathcal{A}_i is an attribute name, and $\mathcal{D} \subset \text{dom}(\mathcal{A}_i)$. A particular case of a simple constraint is the *nil constraint*, defined by $\mathcal{A}_i \in \text{dom}(\mathcal{A}_i)$. For a set of objects \mathcal{O} , we denote by \mathcal{O}_γ the subset of objects for which constraint γ holds. Two constraints γ_1, γ_2 are *disjoint* if for all sets of objects \mathcal{O} , $\mathcal{O}_{\gamma_1} \cap \mathcal{O}_{\gamma_2} = \phi$.

The definition above is a generalization of the normal comparison operators, in that every comparison of an attribute to a value can be re-written in the form of a simple constraint. For the rest of the paper, we will abuse notation and use the usual comparison operators to denote simple constraints.

Definition 4. (*Constraint probability*). A constraint probability is a pair (p, γ) , where p is a rational number in the interval $[0, 1]$ and γ is a conjunction of simple constraints. (p, γ) is called a *nil constraint probability* when γ is true. We will denote these probabilities by (p, true) .

Example 1. For example, in Figure 1, $(0.6, \textit{luminosity} > 0.8)$ is a constraint probability. The 0.3 (or $(0.3, \textit{true})$) label between nodes **orange** and **stars** is a nil constraint probability. Intuitively, the constraint probability $(0.6, \textit{luminosity} > 0.8)$ for the edge **dwarf** \Rightarrow **small** states that for objects classified as **small** that have *luminosity* > 0.8, there is a 0.6 probability that these are in fact **dwarf**. Thus, additional information allows us to refine the default probability of 0.5 for the respective edge. We will now discuss the concept of *labeling* for the edges of a CPO. In the following definition, we denote by $E(G)$ the set of edges for a graph G .

Definition 5. (*Labeling*). Suppose G is a graph whose set of edges is $E(G)$, and suppose Γ is the set of all possible constraint probabilities associated with some set of attributes. A labeling of G is any mapping $\wp : E(G) \rightarrow 2^\Gamma$. As usual, 2^Γ is the powerset of Γ .

Definition 6. (*Valid labeling*). A labeling function \wp with respect to a graph G , a relation \mathcal{R} and an attribute $\mathcal{A} \in \mathcal{R}$ is called a **valid labeling** if:

- (V1) $\forall \text{ edges } (c, d) \in E(G), \forall (p, \gamma) \in \wp(c, d), \gamma \text{ refers only to attributes in } \mathcal{R} - \{\mathcal{A}\};$
- (V2) $\forall \text{ edges } (c, d) \in E(G), \text{ there is exactly one nil constraint probability in } \wp(c, d).$

A valid labeling maps each edge of a graph to a set of constraint probabilities. The set of constraint probabilities for an edge $c \Rightarrow d$ contains only one nil constraint probability – as this represents the probability of an object in d being in c , it must be unique. Intuitively, \mathcal{A} will be the attribute whose domain of values is represented by the ontology, and thus we require that any simple constraints do not refer to the value of \mathcal{A} itself.

We are now ready to define a CPO based on the concepts of constraint probabilities and labeling. A CPO can be informally defined as a POB-schema with edges labeled by multiple constraint probabilities. The valid labeling property will ensure that we have at least one nil constraint probability on every edge. Thus, a CPO is a generalization of a POB-schema.

Definition 7. (*Constraint Probabilistic Ontology*). A CPO with respect to a relation \mathcal{R} and an attribute $\mathcal{A} \in \mathcal{R}$ is a quadruple $(\mathcal{C}, \Rightarrow, \mathbf{me}, \wp)$ where:

- (1) \mathcal{C} is a finite set of classes. \mathcal{C} is a superset of the domain of \mathcal{A} . We should note that a CPO can be defined in such manner as to be associated with more than one attribute of a relation. This is in fact the case with our implementation. However, for the purpose of simplicity the discussion will assume that a CPO is associated with only one attribute within a relation.
- (2) \Rightarrow is a binary relation on \mathcal{C} such that $(\mathcal{C}, \Rightarrow)$ is a directed acyclic graph. The relation $c_1 \Rightarrow c_2$ says that the class c_1 is an immediate subclass of c_2 .
- (3) \mathbf{me} maps each class c to a partition of the set of all immediate subclasses of c .
- (4) \wp is a valid labeling with respect to the graph $(\mathcal{C}, \Rightarrow)$, relation \mathcal{R} and attribute \mathcal{A} .

Definition 8. (*CPO enhanced relation*). A CPO enhanced relation is a triple (T, \mathcal{R}, f) , where T is a set of CPOs, \mathcal{R} is a relation with schema $(\mathcal{A}_1, \dots, \mathcal{A}_n)$ and $f : \{\mathcal{A}_1, \dots, \mathcal{A}_n\} \rightarrow T \cup \{\perp\}$ is a mapping called the mapping function such that:

- $\forall S \in T, \exists j$ such that $f(\mathcal{A}_j) = S$. Intuitively, this means that any CPO in T has at least one attribute it is associated with.
- $\forall S \in T$, suppose $f(\mathcal{A}_i) = S$. We then require that $\text{dom}(\mathcal{A}_i) \subseteq \mathcal{C}_S$. Intuitively, this means that any CPO is well defined with respect to the relation and all attributes it is associated with.

Example 2. Suppose S denotes the CPO in Figure 1. Then $(\{S\}, \text{Stars}, f)$ is a CPO-enhanced relation, with $f(\text{star_type}) = S$ and for any other attribute $\mathcal{A} \in \text{Stars}, \mathcal{A} \neq \text{star_type}, f(\mathcal{A}) = \perp$. The conditions above are satisfied as there is an attribute that maps to S and S is well defined with respect to Stars and the star_type attribute.

Definition 9. (*CPO^k*). A k -order CPO (or CPO^k for short) is a CPO such that for all edges (c, d) in the CPO, $|\wp(c, d)| \leq k$. In particular, a CPO¹ is a simple POB.

Intuitively, a k -order CPO is one that associates at most k constraint probabilities with each edge.

Definition 10. (*Probability path*). For a CPO $\mathbf{S}=(\mathcal{C}, \Rightarrow, \mathbf{me}, \wp)$, we say that two classes $c, d \in \mathcal{C}$ have a path of probability q (written $c \rightsquigarrow_q d$) if $c \Rightarrow^* d$ through c_1, \dots, c_k and there is a function g defined on the set of edges in the c, c_1, \dots, c_k, d chain, such that $g(x, y) \in \wp(x, y) \forall (x, y)$ edges along the chain (i.e. g selects one constraint probability from each edge along the path) and $\prod_{g(x,y)=(p,\gamma)} p \geq q$. We also denote by $\Gamma_g(c \rightsquigarrow_q d)$ the set of constraints selected by g along the path.

We will often abuse notation and just use (T, \mathcal{R}) to denote a CPO-enhanced relation, leaving f implicit.

4 CONSISTENCY

Intuitively, a CPO is consistent if for any universe of objects \mathcal{O} , we can find an assignment of objects to the classes of the CPO that satisfy the conditions required by subclassing and constraint probabilities. As proven in [1], consistency checking for POBs is NP-complete, thus we expect a similar result for CPOs.

Definition 11. (*Consistent CPO*). Let $\mathbf{S}=(\mathcal{C}, \Rightarrow, me, \wp)$ be a CPO. An interpretation of \mathbf{S} is a mapping ϵ from \mathcal{C} to the set of all finite subsets of a set \mathcal{O} (an arbitrary universe of objects). An interpretation ϵ of \mathbf{S} is called a taxonomic probabilistic model if and only if:

- (C1) $\epsilon(c) \neq \phi, \forall c \in \mathcal{C}$;
- (C2) $\epsilon(c) \subseteq \epsilon(d), \forall c, d \in \mathcal{C}, c \Rightarrow d$
- (C3) $\epsilon(c) \cap \epsilon(d) = \phi, \forall c, d \in \mathcal{C}, c \neq d$ that belong to the same cluster $\mathcal{L} \in \cup me(\mathcal{C})$.
- (C4) \forall edges $u \Rightarrow v \in (\mathcal{C}, \Rightarrow), \forall$ constraint probabilities $(p, \gamma) \in \wp(u, v), |\epsilon(u)| \geq p * |\epsilon(v)|_\gamma$, where by $|\epsilon(v)|_\gamma$ we denote the number of objects assigned to class v that meet constraint γ . All objects in a class meet the nil constraint.

A CPO is consistent if and only if it has a taxonomic probabilistic model.

Theorem 1. *The problem of deciding whether a given CPO \mathbf{S} is consistent is NP-complete.*

The problem is NP-complete since the NP-complete problem of deciding whether a weight formula is satisfiable in a measurable probability structure [2] can be polynomially reduced to consistency checking for a CPO. Furthermore, a non-deterministic algorithm that performs consistency checking in polynomial time can be devised.

Nonetheless, polynomial time algorithms for deciding the consistency of a CPO in relevant special cases may be possible. Eiter et. al. [1, 3] introduced the concept of a pseudoconsistent POB — we extend their definition to the case of a CPO and then present a notion of well structured CPOs. A pseudoconsistent and well structured CPO is guaranteed to be consistent. We will introduce the concept of a pseudoconsistent CPO that encompasses most of the properties for pseudoconsistent and well structured POB schemas. Furthermore, we will define our own concept of well-structuredness.

Definition 12. (*Pseudoconsistent CPO*). For a CPO $S = (\mathcal{C}, \Rightarrow, me, \wp)$, we denote $\mathbf{S}^*(c) = \{d \in \mathcal{C} | d \Rightarrow^* c\}$. S is pseudoconsistent if (rule labeling follows the one in [1]):

- (P2) For all classes $c \in \mathcal{C}$ and for all clusters $\mathcal{L} \in me(c)$, no two distinct classes $c_1, c_2 \in \mathcal{L}$ have a common subclass.
- (W1) The $(\mathcal{C}, \Rightarrow)$ graph has a top (or root) element.

- (W2) For every class $c \in \mathcal{C}$, any two distinct immediate subclasses have either no common subclass or a greatest common subclass which is different from them.
- (W3) For every class $c \in \mathcal{C}$, the undirected graph $G_S(c) = (\nu, \epsilon)$ that is defined by $\nu = \text{me}(c)$ and $\epsilon = \{\{\mathcal{L}_1 \times \mathcal{L}_2\} \in \nu \times \nu \mid \mathcal{L}_1 \neq \mathcal{L}_2, \bigcup \mathbf{S}^*(\mathcal{L}_1) \cap \bigcup \mathbf{S}^*(\mathcal{L}_2) \neq \phi\}$ is acyclic (i.e., for every class $c \in \mathcal{C}$, the partition clusters in $\text{me}(c)$ are not cyclically connected through common subclasses). In short, multiple inheritance does not cyclically connect partition clusters.
- (W4) $\forall c \in \mathcal{C}$, if the graph $G_S(c)$ has an edge (i.e. two distinct clusters in $\text{me}(c)$ have a common subclass), then every path from a subclass of c to the top element of $(\mathcal{C}, \Rightarrow)$ goes through c . In short, multiple inheritance can be locally isolated in the graph $(\mathcal{C}, \Rightarrow)$.

Finding a method that checks for consistency in polynomial time requires that we “summarize” the edge labeling information. Even though we lose information through this process, we can determine a subset of CPOs that allow for a reasonable consistency checking algorithm.

Definition 13. (Weight factor). Let P be a set of non-nil constraint probabilities. Then we define a weight factor $w_f(P)$ as follows:

- If $P = \phi$, then $w_f(P) = 0$;
- If $P = \{(p, \gamma)\}$, then $w_f(P) = p$;
- Otherwise, we define w_f recursively with the formula: $w_f(P \cup Q) = w_f(P) + w_f(Q) - w_f(P) \cdot w_f(Q)$.

Lema 1 Let \mathcal{O} be an arbitrary set of objects and let $(p_1, \gamma_1), (p_2, \gamma_2)$ be two non-nil constraint probabilities. Then we can always select a set \mathcal{O}' with $|\mathcal{O}'| \leq (p_1 + p_2 - p_1 \cdot p_2) \cdot |\mathcal{O}|$ such that:

- \mathcal{O}' contains at least $p_1 \cdot |\mathcal{O}_{\gamma_1}|$ objects that meet constraint γ_1 (\mathcal{O}_{γ_1} is the set of objects from \mathcal{O} that meet γ_1);
- \mathcal{O}' contains at least $p_2 \cdot |\mathcal{O}_{\gamma_2}|$ objects that meet constraint γ_2 ;

Theorem 2. Given a **consistent** CPO $\mathbf{S} = (\mathcal{C}, \Rightarrow, \text{me}, \wp)$ and $c \Rightarrow d$ an edge in $(\mathcal{C}, \Rightarrow)$, we write $\wp(c, d) = \{(p_0, \text{true})\} \cup \wp'(c, d)$ (i.e. we isolate the nil constraint probability). Then condition (C4) of Definition 11 for $c \Rightarrow d$ can be satisfied by assigning $w(c, d) \doteq \max(p_0, w_f(\wp'(c, d)))$ objects from class d into class c . (Note: this condition can be satisfied by selecting fewer objects in some instances, but as mentioned before we are trying to identify a subset of CPO viable for consistency checking).

The proof is based on induction on the size of $\wp'(c, d)$.

Definition 14. (Well structured CPO). A CPO $\mathbf{S} = (\mathcal{C}, \Rightarrow, \text{me}, \wp)$ is well-structured if the following conditions hold:

- (W5) $\forall d, c_1, c_2 \in \mathcal{C}$ such that $c_1 \Rightarrow d, c_2 \Rightarrow d, c_1 \neq c_2$ and c_1, c_2 are in the same cluster $\mathcal{L} \in \text{me}(d)$ and $\forall (p_1, \gamma_1) \in \wp(c_1, d), \forall (p_2, \gamma_2) \in \wp(c_2, d)$, if $\gamma_1, \gamma_2 \neq \text{true}$, then γ_1 and γ_2 are **disjoint**.
- (W6) $\forall d \in \mathcal{C}, \forall \mathcal{L} \in \text{me}(d), \forall c \in \mathcal{L}, \sum_c w(c, d) \leq 1$.

Theorem 3. *A well structured and pseudoconsistent CPO is consistent.*

The proof is immediate. Conditions of pseudoconsistency account for (C1)–(C3) in Definition 11, as shown in [1]. Conditions (W5) and (W6) account for (C4) in Definition 11.

We can now give an algorithm for consistency checking. Such an algorithm decides on consistency by checking that a CPO belongs to the set defined in the previous theorem. We will use only such CPOs for the discussions and experiments to follow. The Consistency-Check(S) algorithm is two fold: the first part determines whether S is pseudoconsistent according to Definition 12. This algorithm is the well-structured(S) algorithm defined for POB schemas, complemented with a simple operation to determine the top (if any) of a CPO schema. We can simply reuse this algorithm since it only deals with properties involving inheritance and not with edge labeling. The well-structured(S) algorithm runs in $\mathcal{O}(n^2e)$ time, where $n = |\mathcal{C}|$ and e is the number of directed edges in $(\mathcal{C}, \Rightarrow)$. We will now define a new algorithm that checks for conditions (W5) and (W6). The algorithm is described below.

Algorithm: well-structured-CPO(S)

Input: Pseudoconsistent CPO $\mathbf{S}=(\mathcal{C}, \Rightarrow, \text{me}, \wp)$.

Output: *true* if \mathbf{S} is well-structured and *false* otherwise.

Notation: We denote by $\text{top}(\mathbf{S})$ the top element of $(\mathcal{C}, \Rightarrow)$. Q is a FIFO queue, initially empty.

- 1) $Q \leftarrow \{\text{top}(\mathbf{S})\};$
- 2) **while** $\neg \text{empty}(Q)$ **do**
- 3) $c \leftarrow \text{dequeue}(Q);$
- 4) **for each** $\mathcal{L} \in \text{me}(c)$ **do**
- 5) $Q \leftarrow \text{enqueue}(\mathcal{L}, Q);$ (add all elements within \mathcal{L} to the queue)
- 6) **for each** $d_1, d_2 \in \mathcal{L}, d_1 \neq d_2$ **do**
- 7) **for each** $(p_1, \gamma_1), (p_2, \gamma_2) \in \wp(d_1, c) \times \wp(d_2, c), \gamma_1, \gamma_2 \neq \text{nil}$ **do**
- 8) **if** γ_1, γ_2 are not disjoint, **then**
- 9) **return false;** (\mathbf{S} does not satisfy (W5))
- 10) **end;**
- 11) **end;**
- 12) $\text{Sum} \leftarrow 0;$
- 13) **for each** $d \in \mathcal{L}$ **do**
- 14) Compute $w(d, c);$
- 15) $\text{Sum} \leftarrow \text{Sum} + w(d, c).$
- 16) **end;**
- 17) **if** $\text{Sum} > 1$ **then return false;** (\mathbf{S} does not satisfy (W6))
- 18) **end;**

- 19) **end**;
 20) **return true** (**S** is a well structured CPO);

It is easy to see that the algorithm described here runs in $\mathcal{O}(n^2k^2)$ for a CPO^k . The **for** statements on lines 6 and 7 account for this limit. Computation of $w(d, c)$ is done in $\mathcal{O}(k)$ for every class d , according to the recursive definition given.

So far, we have extended the POB-schema defined by Eiter et. al. [1] in several ways. First, we have added constrained probabilities to the ontology. Second, we have generalized the well-structuredness property, by allowing different paths between two classes to have different probabilities. In the following sections, we introduce new concepts that allow us to extend the relational algebra to handle CPOs.

5 CPO merging

Suppose we wish to perform a natural join operation on two relations, $\mathcal{R} = \mathcal{R}_1 \Theta_{\mathcal{A}} \mathcal{R}_2$. Suppose $\mathcal{R}_1.\mathcal{A}$ is associated with CPO S_1 and $\mathcal{R}_2.\mathcal{A}$ is associated with CPO S_2 . As each attribute has only one associated CPO, we need a method to integrate S_1 and S_2 . We will declaratively define what the integration of two ontologies is, and then provide an algorithm to implement it. The notion of an interoperation constraint specifies any known relationships between terms in the two ontologies.

Definition 15. (*Interoperation constraints*). For the purpose of merging two CPOs $S_1 = (\mathcal{C}_1, \Rightarrow_1, me_1, \wp_1)$, $S_2 = (\mathcal{C}_2, \Rightarrow_2, me_2, \wp_2)$, a **set of interoperation constraints** will contain elements of the following type:

- (*Equality constraint*). $c_1 := c_2$, where $c_1 \in \mathcal{C}_1$ and $c_2 \in \mathcal{C}_2$. This type of constraint specifies that two classes from different CPOs always refer to the same set of objects. In order to avoid situations where by transitivity of the equality constraint we would have two classes from the same CPO in an equality relation, we require that each class from each CPO appear at most once in the set of equality constrains.
- (*Immediate subclassing constraint*). Such constraints are in the form $c_1 \sqsubset_{(p_0, true), (p_1, \gamma_1), \dots, (p_k, \gamma_k)} c_2$, where $c_1 \in \mathcal{C}_1$, $c_2 \in \mathcal{C}_2$ and $(p_0, true), (p_1, \gamma_1), \dots, (p_k, \gamma_k)$ are constraint probabilities such that there is exactly one nil constraint probability. This constraint states that c_1 should be an immediate subclass of c_2 in the merged ontologies with the given labeling on the edge between the two.

The subclassing constraints specify a relation of *immediate subclassing*. This is done firstly in order to simplify the merge algorithm. Secondly, in the extended version of the paper available at <http://www.cs.umd.edu/users/udrea/ProbOntologies.pdf> we provide algorithms to infer equality constraints and thus immediate subclassing constraints would be used when provided by a DB administrator.

Definition 16. (*Integration witness*). Given two CPOs, $S_1 = (\mathcal{C}_1, \Rightarrow_1, me_1, \wp_1)$, $S_2 = (\mathcal{C}_2, \Rightarrow_2, me_2, \wp_2)$ and a set of interoperation constraints \mathcal{I} defined on S_1, S_2 , we define a witness to the integration of S_1, S_2 as $S \doteq \mu_{\mathcal{I}}(S_1, S_2)$ if the following conditions hold:

- (O1) S is a CPO $(\mathcal{C}, \Rightarrow, me, \wp)$. This states that the result of merging two CPOs is a CPO, thus abiding by Definition 7.
- (O2) $\forall c \in \mathcal{C}_1 \cup \mathcal{C}_2$, only one of the following holds: either $c \in \mathcal{C}$ or $\exists d \in \mathcal{C}_1 \cup \mathcal{C}_2$ such that $d \in \mathcal{C}$ and $(c := d) \in \mathcal{I}$. This condition states that all classes from both ontologies would be present in the result, either directly or through a representative if they are part of an equality constraint. We will denote by $CR(x)$ the representative of class x in \mathcal{C} .
- (O3) $\forall c_1 \Rightarrow_1 d_1 \in (\mathcal{C}_1, \Rightarrow_1)$, $c_2 \Rightarrow_2 d_2 \in (\mathcal{C}_2, \Rightarrow_2)$, $CR(c_1) \Rightarrow CR(d_1), CR(c_2) \Rightarrow CR(d_2) \in (\mathcal{C}, \Rightarrow)$. Intuitively, this states that no edges are lost during the merging process.
- (O4) $\forall c_1 \Rightarrow_1 d_1 \in (\mathcal{C}_1, \Rightarrow_1)$, $c_2 \Rightarrow_2 d_2 \in (\mathcal{C}_2, \Rightarrow_2)$ such that $\{c_1 := c_2, d_1 := d_2\} \subseteq \mathcal{I}$, and $\forall (p_1, \gamma_1) \in \wp_1(c_1, d_1)$ and $(p_2, \gamma_2) \in \wp_2(c_2, d_2)$:
 - (O4.1) If $\gamma_1 \neq \gamma_2$, then $(p_1, \gamma_1) \in \wp(CR(c_1), CR(d_1))$ and $(p_2, \gamma_2) \in \wp(CR(c_1), CR(d_1))$.
 - (O4.2) If $\gamma_1 = \gamma_2$, then $\exists! p \in [\min(p_1, p_2), \max(p_1, p_2)]$ such that $(p, \gamma_1) \in \wp(CR(c_1), CR(d_1))$.

This rule gives guidelines for solving conflicts between edges present in both ontologies; the interesting case occurs when two constraint probabilities from the same edge in the different CPOs have the same constraint. In such cases, the constraint appears in S with a probability value between the two probabilities on the conflicting edges.
- (O5) Every $c_1 \sqsubset_{(p_0, true), (p_1, \gamma_1), \dots, (p_k, \gamma_k)} c_2 \in \mathcal{I}$ induces an edge between $CR(c_1)$ and $CR(c_2)$ in $(\mathcal{C}, \Rightarrow)$, labeled with $(p_0, true), (p_1, \gamma_1), \dots, (p_k, \gamma_k)$. If the induced edge conflicts with another edge, then rule (O4) applies. Otherwise, the induced edge is part of the graph $(\mathcal{C}, \Rightarrow)$. Intuitively, this rule states that immediate subclassing constraints induce new edges in S and they are treated as standard edges in case of conflict.
- (O6) $\forall c \in \mathcal{C}_1 \cup \mathcal{C}_2$, $\forall \mathcal{L} \in me_{1|2}(c)$ (me_1 or me_2 is applied depending on where c is from), $\exists! \mathcal{L}' \in me(CR(c))$ such that $\mathcal{L} \subseteq \mathcal{L}'$. Intuitively, this rule tells us that a disjoint decomposition cannot be split as a result of merging.
- (O7) S is pseudoconsistent and well-structured according to Definitions 12 and 14. Intuitively, this rule states that the resulting CPO is consistent and this can be verified in polynomial time.

There can be zero, one or many witnesses to the integration of two CPOs. Two CPOs are said to be *unmergeable* if no witnesses exist. We now present an algorithm that takes two consistent CPOs and either returns a witness to the integration of the two if such a witness exists or throws an exception.

Algorithm: CPOmerge(S_1, S_2, \mathcal{I})

Input: $S_1 = (\mathcal{C}_1, \Rightarrow_1, me_1, \wp_1)$, $S_2 = (\mathcal{C}_2, \Rightarrow_2, me_2, \wp_2)$ – consistent CPOs, interoperation constraint set \mathcal{I} .

Output: $S = \mu_{\mathcal{I}}(S_1, S_2)$ or throws *cannot_merge_exception*.

Notation: We will use *union-find* semantics [4] for the first part of the algorithm. We will denote by $CR(x)$ the *union-find* class representative for element x and we will assume that *unify* is written such that $CR(x)$ will always be in \mathcal{C}_1 . We will regard $[\Rightarrow]$ and $[me]$ as data structures, although they are functions (the parentheses are used for notation purposes only). We will initialize these structures for each of the corresponding function's domain values.

```
(* INITIALIZATION *)
1)  $\mathcal{C} = \mathcal{C}_1$ ;
2)  $[\Rightarrow] = [\Rightarrow_1]$ ;
3)  $[me] = [me_1] \cup [me_2]$ ;
(* ADDING NODES FROM  $S_2$  *)
4) for each  $c_2 \in \mathcal{C}_2$  do
5)   if  $\exists (c_1 := c_2) \in \mathcal{I}$  then
6)     unify( $c_1, c_2$ );
7)   else
8)      $\mathcal{C} = \mathcal{C} \cup \{c_2\}$ ;
9)   end;
10) end;
(* CLEARING UP me *)
11) replace all classes  $x$  from all clusters within me with  $CR(x)$ ;
12) for each  $c \in \mathcal{C}$  do
13)   for each clusters  $\mathcal{L}_1, \mathcal{L}_2 \in me(c)$ ,  $\mathcal{L}_1 \neq \mathcal{L}_2$ , do
14)     for each  $(c_1, c_2) \in \mathcal{L}_1 \times \mathcal{L}_2$  do
15)       if  $CR(c_1) = CR(c_2)$  then
16)          $[me](c) = [me](c) - \{\mathcal{L}_1, \mathcal{L}_2\} \cup \{\mathcal{L}_1 \cup \mathcal{L}_2\}$ ;
17)         break;
18)       end;
19)     end;
20)   end;
21) end;
(* ADDING NEW EDGES *)
22) for each  $(c, d)$  edge in  $S_2$  or subclassing constraint in  $\mathcal{I}$  do
23)   if  $CR(c) \Rightarrow CR(d) \notin [\Rightarrow]$  then
24)      $[\Rightarrow] = [\Rightarrow] \cup (c, d)$ ;
25)     set  $\wp(c, d)$  to  $\wp_2(c, d)$  or the label of the subclassing constraint;
26)   else
27)      $e = [\Rightarrow](c, d)$ ;
28)      $e_1 =$  the second edge  $(c, d)$ 
28') (from subclassing constraint or  $S_2$ ).
29)     for each  $(p, \gamma) \in \wp_2^*(e_1)$  do
29') ( $\wp_2^*$  means either  $\wp_2$  or the the subclassing constraint label)
30)     if  $\exists (p_1, \gamma) \in \wp(e)$  then
31)        $\wp(e) = \wp(e) - \{(p_1, \gamma)\} \cup \{(min(p, p_1), \gamma)\}$ ;
32)        $\wp_2^*(e_1) = \wp_2^* - \{(p, \gamma)\}$ ;
```

```

33)   end;
34)   end;
35)    $\wp(e) = \wp(e) \cup \wp_2^*$ ;
36)   end;
37) end;
38) if not Consistency-Check( $S$ ) then throw cannot_merge_exception;
39) return  $S$ ;

```

CPOmerge starts by using the class hierarchy of one of the two CPOs being merged. We can then use any of the numerous well-known *union-find* algorithms (Tarjan [4]) to add classes from the second ontology to the merge result. Since every class in the two ontologies appears at most once in the equality constraints, the size of each *union-find* set is at most 2, which means that there is almost no time penalty for using *union-find*.

The next step is needed to generate the new clustering function me for the resulting ontology. The property enforced here is: if we have two pairs of classes $c_1 := c_2$ and $d_1 := d_2$ and $c_1 \Rightarrow d_1$ and $c_2 \Rightarrow d_2$, then all siblings of c_2 need to be in the same disjoint decomposition as the siblings of c_1 . Note that this “cluster contraction” process does not need to be recursively repeated. If that were the case, we would be contracting different clusters from the same ontology, which would lead to an inconsistent merge result.

The next step implies adding the edges from the second ontology, as well as those implied by the subclassing constraints. Note that in case of conflict, we always take the minimum probability of the two constraint probabilities. Intuitively, this tends to minimize $w(c, d)$ and thus lead to a consistent merge result. The last step implies checking for consistency. If the resulting CPO is

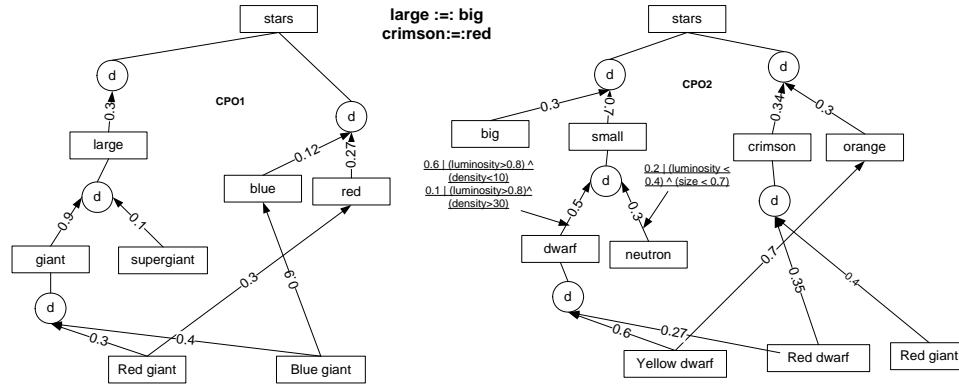


Fig. 2. Merging example

not consistent, then the merge operation fails. There are a number of techniques

that can extend the merge operation to recover from inconsistencies, using loop elimination for instance. These are beyond the scope of this paper. Using the algorithm above, we can extend the merge operation to an arbitrary number of CPOs, by performing merging in pairs. The algorithm above is bounded by $\mathcal{O}(n^3)$, given by the loops involved in restructuring me. The two ontologies in Figure 2 can be merged into the ontology in Figure 1 using the rules depicted.

Theorem 4. (*CPOmerge correctness*).

- (i) $CPOmerge(S_1, S_2, \mathcal{I})$ returns a witness iff there exists a witness to the integrability of S_1, S_2 under interoperation constraints \mathcal{I} .
- (ii) $CPOmerge$ is guaranteed to terminate.

The proof is based on the structure of the algorithm that directly corresponds to conditions (O1)-(O7). Also, the choice of probability in cases of conflict is conservative.

6 CPO OPERATIONS

We now extend the relational algebra to accommodate CPO-enhanced relations.

6.1 Selection

Definition 17. (*Simple condition*). Suppose \mathcal{C} is the set of classes in the CPO associated with relation \mathcal{R} with attributes $\{A_1, \dots, A_m\}$. Then: a simple condition is of the form $A_i \underline{in} T$, where $T \subseteq dom(\mathcal{A}_i) \subseteq \mathcal{C}$.

Example 3. The definition states that in the context of a CPO, simple conditions are those that refer to the attributes represented in the CPO. This is very similar to Definition 3 for simple constraints. However, simple constraints can refer to all attributes of a relation *except* those associated with the current CPO, while a simple condition refers *only* to the attributes associated with the CPO. For instance, in the example of Figure 1, $luminosity > 0.8$ is a simple constraint, but not a simple condition since the CPO does not contain values for the *luminosity* attribute. On the other hand, $star_type \underline{in} \{\text{dwarf}, \text{orange}\}$ is a simple condition, but cannot be a simple constraint, since it refers to the attribute associated with the CPO.

Definition 18. (*Probability condition*). A probability condition for selection is of the form (δ, p) , where δ is a simple condition and p is a rational number in the $[0, 1]$ interval.

Intuitively, a probability condition ensures that only those tuples which satisfy δ with probability p or more will be returned.

Definition 19. (*Selection condition*). A selection condition is any combination of probability conditions or simple conditions using the \vee , \wedge and \neg logical operators.

Example 4. Intuitively, the selection conditions we are interested in are combinations of simple and probability conditions and thus refer only to the attributes associated with a CPO. All other selection operations can be reduced to classical relational algebra. For example,

$$(\text{star_type } \underline{\text{in}} \{\text{dwarf, giant}\}, 0.5) \wedge \neg(\text{star_type } \underline{\text{in}} \{\text{blue}\})$$

is a selection condition.

We now give an extension for the relational selection operation. This definition uses the concept of probability path defined in Section 3.

Definition 20. For a CPO-enhanced relation defined by the CPO $\mathbf{S} = (\mathcal{C}, \Rightarrow, \text{me}, \wp)$ and the relation $\mathcal{R} = \{A_1, \dots, A_m\}$, we define CPO-enhanced selection as follows (Note: As stated above, we are only interested in cases not covered by relational algebra's selection).

- (Simple selection). We define

$$\sigma_{A_i \underline{\text{in}} T}(\mathcal{R}) = \{t \in \mathcal{R} \mid t.A_i \in T \vee \exists c \in T \text{ s.t. } t.A_i \Rightarrow^* c\}.$$

- (Probabilistic selection). We define $\sigma_{(A_i \underline{\text{in}} T, p)}(\mathcal{R}) = \{t \in \mathcal{R} \mid (t.A_i \in T) \vee (\exists c \in T \text{ s.t. } t.A_i \Rightarrow^* c) \vee (\exists c' \in T \text{ s.t. } c' \rightsquigarrow_p t.A_i \text{ and } t.A_i \text{ meets every constraint in } \Gamma(c' \rightsquigarrow_p t.A_i))\}$.

- (Composite selection). For every selection condition of the form $\Delta = \Delta_1 \underline{\text{op}} \Delta_2$, $\sigma_\Delta(\mathcal{R}) = \sigma_{\Delta_1}(\mathcal{R}) \overline{\text{op}} \sigma_{\Delta_2}(\mathcal{R})$. Here, the $\overline{\text{op}}$ operator is the set operation corresponding to the logical operation op. For instance, \cup corresponds to \vee , \cap to \wedge and so on.

Example 5. In the examples of Section 1, the query $\sigma_{(\text{star_type } \underline{\text{in}} \{\text{dwarf}\}, 0.5)}(\text{Stars})$ would return:

- All tuples with $\text{star_type}=\text{dwarf}$;
- All tuples with $\text{star_type} \in \{\text{yellow_dwarf}, \text{red_dwarf}\}$;
- All tuples with $\text{star_type}=\text{small}$. If the probability threshold had been 42%, then we would have also included all tuples that have $\text{star_type}=\text{star}$ and meet the conditions $\text{luminosity} > 0.8$ and $\text{density} < 10$.

6.2 Projection and Cartesian Product

Projection for a CPO-enhanced relation is the same as in relational algebra. An operation that would “trim” the CPO graph is possible, but that would lead to loss of data within the relation, A cartesian product operation on two CPO-enhanced relations will not alter the structure of the respective CPOs, nor the association between the CPOs and the attributes of the two relations. Thus, we reduce the cartesian product operation to its correspondent in classical relational algebra.

6.3 Join

Consider two CPO-enhanced relations $(\mathcal{T}_1, \mathcal{R}_1)$ and $(\mathcal{T}_2, \mathcal{R}_2)$ with mapping functions f_1 and f_2 respectively and suppose their attribute names are renamed so that they share no attributes in common. Suppose $\xi(\mathcal{A}_1, \mathcal{A}_2)$ is a join-condition linking attribute A_1 from \mathcal{R}_1 with attribute A_2 from \mathcal{R}_2 . We say that $(\mathcal{T}_1, \mathcal{R}_1)$ and $(\mathcal{T}_2, \mathcal{R}_2)$ are *join compatible* w.r.t. a finite set \mathcal{I} of interoperability constraints iff the CPOs $f_1(A_1)$ and $f_2(A_2)$ are mergeable w.r.t. \mathcal{I} and join-condition $\xi(\mathcal{A}_1, \mathcal{A}_2)$.

Table 1. Simple relation examples for join

Relation R1			Relation R2		
Name	Density	Star_type	Mass	Location	Star_type
Gliese623a	0.9	red	0.1	Hercules	crimson
Gliese623b	1.2	red_dwarf	143	Leo	blue_giant

Definition 21. (*Join operation*). Suppose $(\mathcal{T}_1, \mathcal{R}_1)$ and $(\mathcal{T}_2, \mathcal{R}_2)$ are join-compatible w.r.t. \mathcal{I} and the join-condition $\xi(\mathcal{A}_1, \mathcal{A}_2)$. Let S be a witness to the integrability of the ontologies associated with $\mathcal{A}_1, \mathcal{A}_2$. Then

- $\mathcal{R} = \mathcal{R}_1 \bowtie_{\xi(\mathcal{A}_1, \mathcal{A}_2)} \mathcal{R}_2$
- $\mathcal{T} = (\mathcal{T}_1 \cup \mathcal{T}_2 \cup \{S\}) - \{S_1, S_2\}$ where S_i is the ontology associated with \mathcal{A}_i .

Table 2. Merge result for $(CPO1, R1) \bowtie_{R1.Star_type=R2.Star_type} (CPO2, R2)$

Name	Density	R1.Star_type	Mass	Location	R2.Star_type
Gliese623a	0.9	red	0.1	Hercules	crimson

Example 6. Given the relations $R1$ and $R2$ in Table 1, assume that the attribute $R1.Star_type$ is mapped to $CPO1$ in Figure 2 and that attribute $R2.Star_type$ is mapped to $CPO2$ in the same figure. Also, we denote by $\mathcal{I}(S1, S2)$ the set of interoperation constraints for the purpose of merging $S1$ and $S2$. \mathcal{I} will contain the equality constraints depicted in Figure 2, as well as any other constraints based on class names that are identical in the two ontologies. Then the result of the join operation $(CPO1, R1) \bowtie_{Star_type} (CPO2, R2)$ is presented in Table 2, with the $R1.Star_type$ and $R2.Star_type$ being mapped to the CPO presented in Figure 1 (here we abbreviate the condition that the $Star_type$ attributes of the two relations should be equal). We obtain this result by first performing the merge between $CPO1$ and $CPO2$ and then checking for the join condition, taking into account any equality constraints inferred.

6.4 Union

Definition 22. (*Union operation*). Suppose $(\mathcal{T}_1, \mathcal{R}_1)$ and $(\mathcal{T}_2, \mathcal{R}_2)$ are CPO-enhanced relations with the same schema. For each attribute A_i , let S_i^1, S_i^2 be the CPO associated with attribute A_i in $(\mathcal{T}_1, \mathcal{R}_1)$ and $(\mathcal{T}_2, \mathcal{R}_2)$ respectively. Let I_i be a set of interoperation constraints. We say $(\mathcal{T}_2, \mathcal{R}_2)$ are CPO-union compatible if S_i^1, S_i^2 have a witness S_i to their integrability under I_i . In this case, we define $(\mathcal{T}_1, \mathcal{R}_1) \cup (\mathcal{T}_2, \mathcal{R}_2)$ as $(\mathcal{T}, \mathcal{R})$ where:

- $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$ Here, \cup denotes the relational algebra union operation.
- $\mathcal{T} = \{S_i \mid A_i \text{ is an attribute of } \mathcal{R}_i\}$.

6.5 Intersection and difference

In order to perform intersection and difference in relational algebra, the relations involved must be union compatible. Therefore, the intersection and difference operations for CPO-enhanced relations are defined similarly to the union operation. The resulting relation \mathcal{R} is either $\mathcal{R}_1 - \mathcal{R}_2$ for difference or $\mathcal{R}_1 \cap \mathcal{R}_2$ for intersection.

7 Related work and conclusions

Wiederhold's group was the first to notice that ontologies can be used to improve the quality of answers to queries. They proposed an *ontology algebra* [5–7]. Their algebras consisted of logical statements [5] using a LISP style syntax. Their later work included graphs which were combined in their ONION system using the three set operations of union, intersection and difference. In addition, a framework called TOSS [8] was recently proposed to increase the recall of queries to XML databases.

In this paper, we present the concept of a probabilistic ontology that builds on the work of Eiter et. al. [1, 3]. We extend the concept of a POB-schema by Eiter et. al. to the case of a constrained probabilistic ontology (CPO) and show how the consistency of POB-schemas can be extended to CPOs. However, our path from there is very different. We show how the answers to *relational queries* can be greatly improved by using probabilities *within the system hidden from users*. We show how multiple CPOs can be combined under a variety of interoperation constraints. Based on these ideas, we develop a relational-style algebra to handle querying with associated ontologies. The interested reader may find initial experimental results, as well as a discussion on how CPOs can be inferred from data sources using a number of different methods in the extended version of the paper available at <http://www.cs.umd.edu/users/udrea/ProbOntologies.pdf>.

8 Acknowledgements

Work supported in part by ARO grant DAAD190310202, NSF grants IIS0329851 and 0205489, and by a DARPA subcontract from the Univ. of California Berkeley.

References

1. Eiter, T., Lu, J.J., Lukasiewicz, T., Subrahmanian, V.: Probabilistic object bases. *ACM Trans. Database Syst.* **26** (2001) 264–312
2. Fagin, R., Halpern, J.Y., Megiddo, N.: A logic for reasoning about probabilities. *Inf. Comput.* **87** (1990) 78–128
3. Eiter, T., Lukasiewicz, T., Walter, M.: A data model and algebra for probabilistic complex values. *Ann. Math. Artif. Intell.* **33** (2001) 105–252
4. Tarjan, R.E.: Efficiency of a good but not linear set union algorithm. *J. ACM* **22** (1975) 215–225
5. Maluf, D., Wiederhold, G.: Abstraction of representation for interoperation. *Lecture Notes in AI, subseries of LNCS* **1315** (1997)
6. Mitra, P., Wiederhold, G., Kersten, M.: A graph-oriented model for articulation of ontology interdependencies. In: *Proceedings Conference on Extending Database Technology, (EDBT'2000)*. (2000) 303–316
7. Wiederhold, G.: Interoperation, mediation and ontologies. In: *International Symp. on Fifth Generation Computer Systems, Workshop on Heterogeneous Cooperative Knowledge Bases*. (1994) 33–48
8. Hung, E., Deng, Y., Subrahmanian, V.S.: Toss: an extension of tax with ontologies and similarity queries. In: *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, ACM Press (2004) 719–730