Query Evaluation and Optimization in the Semantic Web

Edna Ruckhaus, Eduardo Ruiz, and María-Esther Vidal

Dept. of Computer Science and Information Technology
Universidad Simón Bolívar, Venezuela
{ruckhaus,eruiz,mvidal}@ldc.usb.ve

Abstract. We address the problem of answering Web ontology queries efficiently. An ontology is formalized as a *Deductive Ontology Base* (DOB), a deductive database that comprises the ontology's inference axioms and facts, and we present a cost-based query optimization technique for DOB. A hybrid cost model is proposed to estimate the cost and cardinality of basic and inferred facts. Cardinality and cost of inferred facts are estimated using an adaptive sampling technique, while techniques of traditional relational cost models are used for estimating the cost of basic facts and conjunctive ontology queries. Finally, we implement a dynamic-programming optimization algorithm to identify query evaluation plans that minimize the number of intermediate inferred facts. We modeled a subset of the Web ontology language OWL Lite as a DOB, and performed an experimental study to analyze the predictive capacity of our cost model and the benefits of the query optimization technique. Our study has been conducted over synthetic and real-world OWL ontologies, and shows that the techniques are accurate and improve query performance.

1 Introduction

Ontology systems usually provide reasoning and retrieval services that identify the basic facts that satisfy a requirement, and derive implicit knowledge using the ontology's inference axioms. In the context of the Semantic Web, the number of inferred facts can be extremely large. On one hand, the amount of basic ontology facts (domain concepts and Web source annotations) can be considerable, and on the other hand, *Open World* reasoning in Web ontologies may yield a large space of choices. Therefore, efficient evaluation strategies are needed in Web ontology's inference engines.

In our approach, ontologies are formalized as a deductive database called a *Deductive Ontology Base* (DOB). The extensional database comprises all the ontology language's statements that represent the explicit ontology knowledge. The intensional database corresponds to the set of deductive rules which define the semantics of the ontology language. We provide a cost-based optimization technique for Web ontologies represented as a DOB.

© Copyright 2006 for the individual papers by the individual authors. Copying permitted for private and scientific purposes. Re-publication of material in this volume requires permission of the copyright owners.

Traditional query optimization techniques for deductive databases systems include join-ordering strategies, and techniques that combine a bottom-up evaluation with top-down propagation of query variable bindings in the spirit of the Magic-Sets algorithm [17]. Join-ordering strategies may be heuristic-based or cost-based; some cost-based approaches depend on the estimation of the *join selectivity*; others rely on the *fan-out* of a literal [22]. Cost-based query optimization has been successfully used by relational database management systems; however, these optimizers are not able to estimate the cost or cardinality of data that do not exist a priori, which is the case of intensional predicates in a DOB.

We propose a hybrid cost model that combines two techniques for cardinality and cost estimation: (1) the sampling technique proposed in [10, 11] is applied for the estimation of the evaluation cost and cardinality of intensional predicates, and (2) a cost model *à* la System R cost model is used for the estimation of the cost and cardinality of extensional predicates and the cost of conjunctive queries.

Three evaluation strategies are considered for "joining" predicates in conjunctive queries. They are based on the Nested-Loop, Block Nested-Loop, and Hash Join operators of relational databases [16]. To identify a good evaluation plan, we provide a dynamic-programming optimization algorithm that orders subgoals in a query, considering estimates of the subgoal's evaluation cost.

We modeled a subset of the Web ontology language OWL Lite [12] as a DOB, and performed experiments to study the predictive capacity of the cost model and the benefits of the ontology query optimization techniques. The study has been conducted over synthetic and real-world OWL ontologies. Preliminary results show that the cost-model estimates are pretty accurate and that optimized queries are significantly less expensive than non-optimized ones.

Our current formalism does not represent the OWL built-in constructor *ComplementOf*. We stress that in practice this is not a severe limitation. For example, this operator is not used in any of the three real-world ontologies that we have studied in our experiments; and in the survey reported in [23], only 21 ontologies out of 688 contain this constructor.

Our work differs from other systems in the Semantic Web that combine a Description Logics (DL) reasoner with a relational DBMS in order to solve the scalability problems for reasoning with individuals [3, 6, 7, 15]. Clearly, all of these systems use the query optimization component embedded in the relational DBMS; however, they do not develop cost-based optimization for the implicit knowledge, that is, there is no estimation of the cost of data not known a priori.

Other systems use Logic Programming (LP) to reason on large-scale ontologies. This is the case of the projects described in [5, 8, 13] . In Description Logic Programs (DLP) [5], the expressive intersection between DL and LP without function symbols is defined. DL queries are reduced to LP queries and efficient LP algorithms are explored. The project described in [8, 13] reduces a $\mathcal{SHIQ}$ knowledge base to a Disjunctive Datalog program. Both projects apply Magic-Sets rewriting techniques but to the best of our knowledge, no cost-based optimization techniques have been developed. The OWL Lite$^-$ species of the OWL language proposed in [2] is based in the DLP project; it corresponds to the por-

tion of the OWL Lite language that can be translated to Datalog. All of these systems develop LP reasoning with individuals, whereas in the DOB model we develop Datalog reasoning with both, domain concepts and individuals.

In [4], an efficient bottom-up evaluation strategy for HEX-programs based on the theory of *splitting sets* is described. In the context of the Semantic Web, these non-monotonic logic programs contain higher-order atoms and external atoms that may represent RDF and OWL knowledge. However, their approach does not include determining the best evaluation strategy according to a certain cost metric.

In the next section we describe our DOB formalism. Following this, we describe the DOB-S System architecture, Then, we model a subset of OWL Lite as a DOB and present a motivating example. Next, we develop our hybrid cost model and query optimization algorithm. We describe our experimental study and, finally, we point out our conclusions and future work.

2 The Deductive Ontology Base (DOB)

In general, an ontology knowledge base can be defined as:

Definition 1 (Ontology Knowledge Base) *An **ontology knowledge base** **O** is a pair $O = \langle \mathcal{F}, \mathcal{I} \rangle$, where $\mathcal{F}$ is the set of ontology facts that represent the explicit ontology structure (domain) and source annotations (individuals), and $\mathcal{I}$ is the set of axioms that allow the inference of new ontology facts regarding both domain and individuals.*

We will model O as a deductive database which we call a *Deductive Ontology Base* (DOB). A DOB is composed of an Extensional Ontology Base (EOB) and an Intensional Ontology Base (IOB). Formally, a DOB is defined as:

Definition 2 (DOB) *Given an ontology knowledge base $O = \langle \mathcal{F}, \mathcal{I} \rangle$, a **DOB** is a deductive database composed of a set of built-in EOB ground predicates representing $\mathcal{F}$ and a set of IOB built-in predicates representing $\mathcal{I}$, i.e. that define the semantics of the EOB built-in predicates.*

IOB predicates and DOB queries are defined as follows:

Definition 3 (Intensional Predicate) *Given a DOB composed of an EOB and an IOB, an **Intensional Predicate** is a rule $R{:}H(\overline{X}) \leftarrow \exists \overline{Y} B(\overline{X}, \overline{Y})$, where H is the **head**, B is the **body** that corresponds to a conjunction of predicates, and $\overline{X}$ and $\overline{Y}$ are called distinguished variables and non-distinguished variables, respectively. H belongs to the IOB. Predicates in B can belong to the EOB or to the IOB (no negations are allowed).*

Definition 4 (DOB Query) *A **DOB query** is defined as a rule $q : Q(\overline{X}) \leftarrow \exists \overline{Y} B(\overline{X}, \overline{Y})$, where B is the query's goal.*

Next, we provide the definitions related to query-answering for DOBs.

Definition 5 (Valuation) *Given a set of variables $\mathcal{V}$ and a set of constants $\mathcal{C}$, a mapping or valuation γ is a function $\gamma : \mathcal{V} \rightarrow \mathcal{C}$.*

Definition 6 (Valid Instantiation) *Given a Deductive Ontology Base $\mathcal{O}$, a set of constants $\mathcal{C}$ in $\mathcal{O}$, a set of variables $\mathcal{V}$, a rule R, and an interpretation $\mathbb{I}$ of $\mathcal{O}$ that corresponds to its Minimal Perfect Model [1], a valuation γ is a* ***valid instantiation*** *of R if and only if, $\gamma(R)$ evaluates to true in $\mathbb{I}$.*

Definition 7 (Intermediate Inferred Facts) *Given a Deductive Ontology Base $\mathcal{O}$, and a query $q : Q(\overline{X}) \leftarrow \exists \overline{Y} B(\overline{X}, \overline{Y}))$. A proof tree for q wrt $\mathcal{O}$ is defined as follows:*

- *Each node in the tree is labeled by a predicate in $\mathcal{O}$.*
- *Each leaf in the tree is labeled by a predicate in $\mathcal{O}$'s EOB.*
- *The root of the tree is labeled by Q*
- *For each internal node N including the root, if N is labeled by a predicate A defined by the rule R, $A(\overline{X}) \leftarrow \exists \overline{Y} C(\overline{X}, \overline{Y}))$, where $C(\overline{X}, \overline{Y}))$ is the conjunction of the predicates $C_1, ..., C_n$, then, for each valid instantiation of R, γ, the node N has a sub-tree whose root is $\gamma(A(\overline{X}))$ and its children are respectively labeled $\gamma(C1)$,..., $\gamma(C_n)$.*

The valuations needed to define all the valid instantiations in the proof tree correspond to the **Intermediate Inferred Facts** *of q.*

The number of intermediate inferred facts measures the evaluation *cost* of the query Q. Additionally, since the valid instantiations of Q in the proof tree correspond to the answers of the query, the *cardinality* of Q corresponds to the number of such instantiations.

Note that the sets of EOB and IOB built-in predicates of a DOB define an ontology framework, so our model is not tied to any particular ontology language. To illustrate the use of our approach we focus on OWL Lite ontologies.

3 The DOB-S System's Architecture

DOB-S is a system that allows an agent to pose efficient conjunctive queries to ontologies. The system's architecture can be seen in Figure 1.

A subset of a given OWL ontology is translated into a DOB using an OWL Lite to DOB **translator**. EOB and IOB predicates are stored as a deductive database. Next, an **analyzer** generates the ontology's statistics: for each EOB predicate, the analyzer computes the number of facts or valid instantiations in the DOB (cardinality), and the number of different values for each of its arguments (nKeys); for each IOB predicate, an adaptive sampling algorithm [10] is applied to compute cardinality and cost estimates.

When an agent formulates a conjunctive query, the DOB-S system's **optimizer** generates an efficient query evaluation plan. A dynamic-programming optimizer is based in a **hybrid cost model**: it uses the ontology's EOB and IOB statistics, and estimates the cost of a query according the different evaluation strategies implemented. Finally, an **execution engine** evaluates the query plan and produces a query answer.

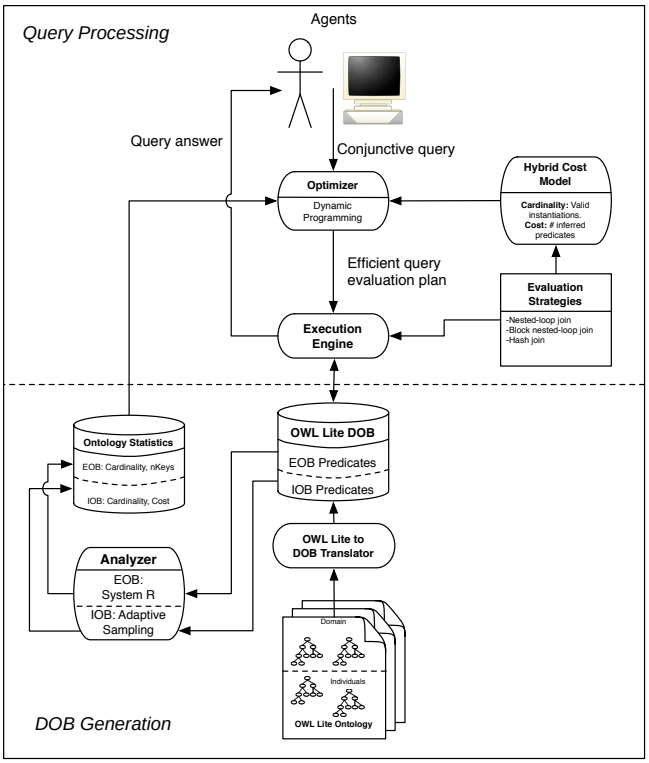


Fig. 1. DOB-S System Architecture

4 OWL Lite DOB

An OWL Lite ontology contains: (1) a set of axioms that provides information about classes and properties, and (2) a set of facts that represents individuals in the ontology, the classes they belong to, and the properties they participate in.

Restrictions allow the construction of class definitions by restricting the values of their properties and their cardinality. Classes may also be defined through the intersection of other classes. Object properties represent binary relationships between individuals; datatype properties correspond to relationships between individuals and data values belonging to primitive datatypes.

The subset of OWL Lite represented as a DOB does not include domain and range class intersection. The `someValuesFrom` restriction is not included as it involves an existential quantifier and cannot be translated to Datalog. Primitive datatypes are not handled; therefore, we do not represent ranges for Datatype properties[1].

[1] `EquivalentClasses`, `EquivalentProperties`, and `allDifferent` axioms, and the `cardinality` restriction are not represented as they are syntactic sugar for other language constructs.

4.1 OWL Lite DOB Syntax

Our formalism, DOB, provides a set of EOB built-in predicates that represents all the axioms and restrictions of an OWL Lite subset.

EOB predicates are *ground*, i.e., no variables are allowed as arguments. A set of IOB built-in predicates represents the semantics of the EOB predicates. We have followed the OWL Web Ontology Language Overview presented in [12].

Table 1 illustrates the EOB and IOB built-in predicates for an OWL Lite subset[2]. Note that some predicates refer to domain concepts (e.g. `isClass, areClasses`), and some to individuals (e.g. `is isIndividual, areIndividuals`).

EOB PREDICATE	**DESCRIPTION**
`isOntology(O)`	An ontology has an Uri `O`
`isImpOntology(O1,O2)`	Ontology `O1` imports ontology `O2`
`isClass(C,O)`	`C` is a class in ontology `O`
`isOProperty(P,D,R)`	`P` is an object property with domain `D` and range `R`
`isDProperty(P,D)`	`P` is a datatype property with domain `D`
`isTransitive(P)`	`P` is a transitive property
`subClassOf(C1,C2)`	`C1` is subclass of `C2`
`AllValuesFrom(C,P,D)`	`C` has property `P` with all values in `D`
`isIndividual(I,C)`	`I` is an individual belonging to class `C`
`isStatement(I,P,J)`	`I` is an individual that has property `P` with value `J`
IOB PREDICATE	**DESCRIPTION**
`areSubClasses(C1,C2)`	`C1` are the direct and indirect subclasses of C2
`areImpOntologies(O1,O2)`	`O1` import the ontologies `O2` directly and indirectly
`areClasses(C,O)`	`C` are all the classes of an ontology and its imported ontologies `O`
`areIndividuals(I,C)`	`I` are the individuals of a class and all of its direct and indirect superclasses `C`; or `I` are the individuals that participate in a property and belong to its domain or range `C`, or are values of a property with all values in `C`

Table 1. Some built-in EOB and IOB Predicates for a subset of OWL Lite

OWL ABSTRACT SYNTAX	EOB PREDICATES
$Ontology(O)$	`isOntology(O)`
$Individual(O1\ value(owl:imports\ O2))$	`impOntology(O1, O2)`
$Ontology(O)$, $Class(C\ partial$	`isClass(C,O)`
$Class(A\ partial\ C)$	`subClassOf(A,C)`
$Class(C1\ partial\ restriction(P\ allValuesFrom(C2)))$	`allValuesFrom(C1,P,C2)`
$Class(A\ partial\ C1 \ldots Cn)$	`subClassOf(A,C1),...,` `subClassOf(A,Cn)`
$ObjectProperty(P\ domain(D))$, $ObjectProperty(P\ range(R))$	`isOProperty(P,D,R)`
$DatatypeProperty(P\ domain(D))$	`isDProperty(P,D)`
$Property(P\ Transitive)$	`isTransitive(P)`
$Individual(I\ type(C))$	`isIndividual(I,C)`
$Individual(I\ value(P\ J))$	`isStatement(I,P,J)`

Table 2. Mapping OWL Lite subset to EOB Predicates

[2] We assume that the class *owl:Thing* is the default value for the domain and range of a property.

OWL LITE INFERENCE RULES	IOB RULE DEFINITIONS

If `subClassOf(C1,C2)` and `subClassOf(C2,C3)` then `subClassOf(C1,C3)`	`areSubClasses(C1,C2):-subClassOf(C1,C2).`
	`areSubClasses(C1,C2):-subClassOf(C1,C3),`
	`areSubClasses(C3,C2).`

If `impOntology(O1,O2)` and `impOntology(O2,O3)` then `impOntology(O1,O3)`	`areImpOntologies(O1,O2):-impOntology(O1,O2).`
	`areImpOntologies(O1,O2):-impOntology(O1,O3),`
	`areImpOntologies(O3,O2).`

If `isClass(C1,O2)` and `impOntology(O1,O2)` then `isClass(C1,O1)`	`areClasses(C,O):-isClass(C,O).`
	`areClasses(C,O1):-isClass(C,O2),`
	`areImpOntologies(O1,O2).`

If `isSubClassOf(C1,C2)` and `isIndividual(I,C1)` then `isIndividual(I,C2)`	`areIndividuals(I,C):-isIndividual(I,C).`
	`areIndividuals(I,C2):-isIndividual(I,C1),`
	`areSubClasses(C1,C2).`

If `isStatement(I,P,J)` and `isOProperty(P,C,R)` then `isIndividual(I,C)`	`areIndividuals(I,C):-isOProperty(P,C,R),` `areStatements(I,P,J).`
If `isStatement(I,P,J)` and `isOProperty(P,D,C)` then `isIndividual(J,C)`	`areIndividuals(J,C): isOProperty(P,D,C),` `areStatements(I,P,J).`
If `isStatement(I,P,J)` and `isDProperty(P,C)` then `isIndividual(I,C)`	`areIndividuals(I,C):-isDProperty(P,C),` `areStatements(I,P,J).`

If `AllValues(C1,P,C)` and `isStatement(I,P,J)` and `isIndividual(I,C1)` then `isIndividual(J,C)`	`areIndividuals(J,C):-isIndividual(I,C1),`
	`allValuesFrom(C1,P,C),`
	`areStatements(I,P,J).`

Table 3. Mapping OWL Lite subset Inference Rules to IOB Predicates

4.2 OWL Lite DOB Semantics

A model-theoretic semantics for an OWL Lite (subset) DOB is as follows:

Definition 8 (Interpretation) *An **Interpretation** $I = (\Delta^I, \mathcal{P}^I, .^I)$ consists of:*

- *A non-empty interpretation domain Δ^I corresponding to the union of the sets of valid URIs of ontologies, classes, object and datatype properties, and individuals. These sets are pairwise disjoint.*
- *A set of interpretations $\mathcal{P}^I$, of the EOB and IOB built-in predicates in Table 1.*
- *An interpretation function $.^I$ which maps each n-ary built-in predicate $p^I \in \mathcal{P}^I$ to an n-ary relation $\prod_{i=1}^{n} \Delta^I$.*

Definition 9 (Satisfiability) *Given an OWL Lite DOB $\mathcal{D}$, an interpretation I, and a predicate $p \in \mathcal{D}$, $I \models p$ iff:*

- *p is an EOB predicate $p(t_1, ..., t_n)$ and $(t_1, ..., t_n) \in p^I$.*
- *p is an IOB predicate $R{:}H(\overline{X}) \leftarrow \exists \overline{Y} B(\overline{X}, \overline{Y})$, and whenever I satisfies each predicate in the body B, I also satisfies the predicate in the head H.*

Definition 10 (Model) *Given an OWL Lite DOB $\mathcal{D}$ and an interpretation I, I is a **model** of $\mathcal{D}$ iff for every predicate $p \in \mathcal{D}$, $I \models p$.*

4.3 Translation of OWL Lite to OWL Lite DOB

A definition of a translation map from OWL Lite to OWL Lite DOB is the following:

Definition 11 (Translation) *Given an OWL Lite theory $\mathcal{O}$ and an OWL Lite DOB theory $\mathcal{D}$, an* ***OWL Lite to DOB Translation*** *$\mathcal{T}$ is a function $\mathcal{T} : \mathcal{O} \rightarrow \mathcal{D}$.*

Given an OWL Lite ontology $\mathcal{O}$, an OWL Lite DOB ontology $\mathcal{D}$ is defined as follows:

- (Base Case) If o is an axiom or fact belonging to the sets of axioms or facts of $\mathcal{O}$, then an EOB predicate $\mathcal{T}(o)$ is defined according to the EOB mappings in Table 2.
- If o is an OWL Lite inference rule, then an IOB predicate $\mathcal{T}(o)$ is defined according to the IOB mappings in Table 3.

The translation ensures that the following theorem holds:

Theorem 1 *Let $\mathcal{O}$ and $\mathcal{D}$ be OWL Lite and OWL Lite DOB theories respectively, and $\mathcal{T}$ be an OWL Lite to DOB Translation such that, $\mathcal{T}(\mathcal{O}) = \mathcal{D}$, then $\mathcal{D} \models \mathcal{O}$.*

5 A Motivating Example

Consider a 'cars and dealers' domain ontology `carsOnt` and Web source ontologies `source1` and `source2`. Source `source1` publishes information about all types of vehicles and dealers, whereas `source2` is specialized in SUVs.

The OWL Lite ontologies can be seen in Table 4.

Ontology carsOnt	**Ontology source1**	**Ontology source2**
`Class vehicle partial Thing)`	`imports carsOnt`	`imports carsOnt`
`SubClassOf(suv,vehicle)`		`individual(s123 type(suv))`
`SubClassOf(car,vehicle)`		
`Property(price domain(vehicle))`		
`Class dealer partial Thing)`		
`Property(sells domain(dealer))`		
`Property(sells range(vehicle))`		
`Property(traction domain(suv))`		
`Property(model domain(vahicle))`		

Table 4. Example OWL Lite ontology

A portion of the example's EOB can be seen in Table 5.

EOB PREDICATES		
`isOntology(carsOnt)`	`isOntology(source1)`	`isOntology(source2)`
`impOntology(source1,carsOnt)`	`impOntology(source2,carsOnt)`	`isClass(vehicle,carsOnt)`
`isClass(vehicle,carsOnt)`	`isClass(dealer,carsOnt)`	`subClassOf(car,vehicle)`
`subClassOf(suv,vehicle)`	`isOProperty(sells,dealer,vehicle)`	`isDProperty(model,vehicle)`
`isDProperty(price,vehicle)`	`isDProperty(traction,suv)`	`isIndividual(s123,suv)`

Table 5. Example DOB ontology

To illustrate a rule evaluation, we will take a query `q` that asks for *the Web sources that publish information about 'traction'*:

```
q(O):-areClasses(C,O),isDProperty(traction,C).
```

The answer to this query corresponds to all the ontologies with classes characterized by the property `traction`, i.e., ontologies `source1`, `source2` and `carsOnt`.

If we invert the ordering of the first two predicates in `q`, we will have an equivalent query `q'`:

```
q'(O):-isDProperty(traction,C),areClasses(C,O).
```

The cost or total number of inferred facts for `q` is larger than the cost for `q'`. In `q`, the number of instantiations or cardinality for the first intensional predicate `areClasses(C,O)` is twelve, four for each ontology, as `source1` and `source2` inherit the classes in `carsOnt`. The cost of inferring these facts is dependent on the cost of evaluating the `areClasses` rule. In `q'`, for the first subgoal `isDProperty(traction,C)`, we have one instantiation: `isDProperty(traction,suv)`. Again, the cost of inferring this fact depends on the cost of the `isDProperty` predicate.

Note that statistics on the size and argument values of the EOB `isDProperty` predicate can be computed, whereas statistics for the IOB `areClasses` predicate will have to be estimated as data is not known a priori. Once the cost of each query predicate is determined, we may apply a cost-based join-ordering optimization strategy.

6 DOB Hybrid Cost Model

The process of answering a query relies on inferring facts from the predicates in the DOB. Our cost metric is focused on the number of intermediate facts that need to be inferred in order to answer the query. The objective is to find an order of the predicates in the body of the query, such that the number of intermediate inferred facts is reduced. We will apply a join-ordering optimization strategy *à* la System R using Datalog-relational equivalences [1]. To estimate the cardinality and evaluation cost of the intensional predicates, we have applied an adaptive sampling technique. Thus, we propose a hybrid cost model which combines adaptive sampling and traditional relational cost models.

6.1 Adaptive Sampling Technique

We have developed a sampling technique that is based on the *adaptive sampling method* proposed by Lipton, Naughton, and Schneider [10, 11]. This technique assumes that there is a population $\mathbb{P}$ of all the different valid instantiations of a predicate P, and that $\mathbb{P}$ is divided into n partitions according to the n possible instantiations of one or more arguments of P. Each element in $\mathbb{P}$ is related to its evaluation cost and cardinality, and the population $\mathbb{P}$ is characterized by the statistics mean and variance.

The objective of the sampling is to identify a sample of the population $\mathbb{P}$, called $\mathbb{EP}$, such that the mean and variance of the cardinality (resp. evaluation cost) of $\mathbb{EP}$ are valid to within a predetermined accuracy and confidence level.

To estimate the mean of the cardinality (resp. cost) of $\mathbb{EP}$, say $\overline{Y}$, within $\frac{\overline{Y}}{d}$ with probability p, where $0 \leq p < 1$ and $d > 0$, the sampling method assumes an *urn* model.

The urn has n balls from which m samplings are repeatedly taken, until the sum z of the cardinalities (resp. costs) of the samples is greater than $\alpha \times (\frac{S}{Y})$, where $\alpha = \frac{d \times (d+1)}{(1-\sqrt{p})}$. The estimated mean of the cardinality (resp. cost) is: $\overline{Y} = \frac{z}{m}$.

The values d and $\frac{1}{(1-\sqrt{p})}$ are associated with the relative error and the confidence level, and S and Y represent the cardinality (resp. cost) variance and mean of $\mathbb{P}$. Since statistics of $\mathbb{P}$ are unknown, the upper bound $\alpha \times \frac{S}{Y}$ is replaced by $\alpha \times b(n)$.

To approximate $b(n)$ for cost and cardinality estimates, we apply Double Sampling [9]. In the first stage we randomly evaluate k samples and take the maximum value among them:

$b(n) = max_{i=1}^{k}(card(P_i))$ (resp. $b(n) = max_{i=1}^{k}(cost(P_i))$), where $1 \leq k \leq n$

It has been shown that a few samples are necessary in order for the distribution of the sum to begin to look normal. Thus, the factor $1/(1-\sqrt{p})$ may be improved by central limit theorem [11]. This improvement allows us to achieve accurate estimations and lower bounds.

Estimating cardinality. Given an intensional predicate P, the *cardinality* of P corresponds to the number of the valid instantiations of P (Definition 6). In our previous example, the number of ontology values obtained in the answer of the query is estimated using this metric.

To estimate the cardinality of P, we execute the adaptive sampling algorithm explained before, by selecting any argument of P, and partitioning $\mathbb{P}$ according to the chosen argument. The cardinality estimation will be $card(P) = \overline{Y} \times n$, where n is the number of partitions, i.e. the number of different instantiations for the chosen argument.

Note that once the cardinality of the non-instantiated P is estimated, we can estimate the cardinality of the instantiated predicate by using the selectivity value(s) of the instantiated argument(s).

Estimating cost. The *cost* of P measures the number of intermediate inferred facts (Definition 7). For instance, to estimate the cost of a predicate $P(X, Y)$, we consider the different instantiation patterns that the predicate can have, i.e., we independently estimate the cost for $P(X^b, Y^b)$, $P(X^b, Y^f)$, $P(X^f, Y^b)$ and $P(X^f, Y^f)$, where b and f indicate that the argument is bound and free, respectively.

The computation of several cost estimates is necessary because in Datalog top-down evaluation [1], the cost of an instantiated intensional predicate cannot

be accurately estimated from the cost of a non-instantiated predicate (using selectivity values). Instantiated arguments will propagate in the IOB rule's body through sideways-passing, and cost varies according to the binding patterns. For example, the cost of `areClasses(C1`^{b}`,C2`^{f}`)` may be smaller than the cost of `areClasses(C1`^{f}`,C2`^{b}`)`, i.e., the bound argument `C1` "pushes" instantiations in the definition of the rule:

```
areSubClasses(C1,C2):-isSubClass(C1,C3),areSubClasses(C3,C2).
```

making its body predicates more selective.

For $P(X^b, Y^b)$, $P(X^b, Y^f)$ and $P(X^f, Y^b)$, we partition $\mathbb{P}$ according to the bound arguments. In these cases we are estimating the cost of one partition. Therefore, $cost(P) = \frac{\overline{Y} \times n}{n} = \overline{Y}$.

Finally, to estimate the cost of $P(X^f, Y^f)$, we choose an argument of P and partition P according to the chosen argument. To reduce the cost of computing the estimate, we choose the most selective argument. The cost estimate is $cost(P) = \overline{Y} \times n$.

Determining the number of partitions n. For both, cost and cardinality estimates, we need to determine the number of possible instantiations, n, of the chosen argument. This value depends on the semantics of the particular predicate. For instance, for an interpretation I, $areClasses(Class, Ont)^I \subseteq \mathcal{C} \times \mathcal{O}$, where $\mathcal{C}$ is the set of valid class URIs and $\mathcal{O}$ is the set of valid ontology URIs. $|\mathcal{C}|$ corresponds to the number of EOB predicates $isClass(Class, Ont)$, i.e. $|\mathcal{C}|$=$Card(isClass(Class, Ont))$. Similarly, $|\mathcal{O}|$=$Card(isOntology(Ont))$; these cardinalities have been computed previously. We assume that the values are uniformly distributed.

6.2 System R Technique

To estimate the cardinality and cost of two or more predicates, we use the cost model proposed in System R. The cardinality of the conjunction of predicates P_1,P_2 is described by the following expression:

$$card(P_1, P_2) = card(P_1) \times card(P_2) \times reductionFactor(P_1, P_2)$$

$reductionFactor(P_1, P_2)$ reflects the impact of the sideways passing variables in reducing the cardinality of the result. This value is computed assuming that sideways passing variables are independent and each is uniformly distributed [18]. For cost estimation, we consider three evaluation strategies:

1. Nested-Loop Join
   Following a Nested-Loop Join evaluation strategy, for each valid instantiation in P_1, we retrieve a valid instantiation in P_2 with a matching "join" argument value:

   $$cost(P_1, P_2) = cost(P_1) + card(P_1) \times cost^{inst}(P_2)$$

   $cost^{inst}(P_2)$ corresponds to the estimate of the cost of the predicate P_2 where the "join" arguments are instantiated in P_2, i.e., all the sideways

passing variables from P_1 to P_2 are bound in P_2. These binding patterns were considered during the sampling-based estimation of the cost of P_2.

2. Block Nested-Loop Join
   Predicate P_1 is evaluated into blocks of fixed size, and then each block is "joined" with P_2.

$$cost(P_1, P_2) = cost(P_1) + \lceil \frac{card(P_1)}{BlockSize} \rceil \times cost(P_2)$$

3. Hash Join
   A hash table is built for each predicate according to their join argument. The valid instantiations of predicates P_1 and P_2 with the same hash key will be joined together:

$$cost(P_1, P_2) = cost(P_1) + cost(P_2)$$

Although the sampling technique is appropiate for estimating a single predicate, it may be inefficient for estimating the size of a conjunction of more than two predicates.

The sampling algorithm in [10] suggests that for a conjunction of 2 predicates, P, Q, if the size of P is n, the query is n-partitionable, that is, for each valid instantiation p in P, the corresponding partition of Q is all the valid instantiations q in Q such that q "joins" p. Therefore, when the size of the first predicate in a query is small, our sample size may be larger. This problem can be extended to conjunctive queries with several subgoals, so when the number of intermediate results is small, sampling time may be as large as evaluation time.

6.3 Query Optimization

In Figure 2 we present the algorithm used to optimize the body of a query. The proposed optimization algorithm extends the System R dynamic-programming algorithm by identifying orderings of the n EOB and IOB predicates in a query. During each iteration of the algorithm, the best intermediate sub-plans are chosen based on cost and cardinality. In the last iteration, final plans are constructed and the best plan is selected in terms of the cost metric.

During each iteration i between 2 and *n-1*, different orderings of the predicates are analyzed. Two subplans are considered equivalents if and only if, they are composed by the same predicates. A subplan SP_i is better than a subplan SP_j if and only if, the cost and cardinality of SP_j are greater than cost and cardinality of SP_i. If SP_i cost is greater than SP_j cost, but SP_j cardinality is greater than SP_i cardinality, i.e. they are un-comparable, then the equivalence class is annotated with the two subplans.

7 Experimental Results

An experimental study was conducted for synthetic and real-world ontologies. Experiments on synthetic ontologies were executed on a SunBlade 150 (650MHz)

Algorithm Dynamic Programming
INPUT: `Predicate`: a set of predicates, P_1,...,P_n.
OUTPUT: `OrderedPredicate`: an ordering of `Predicate`

1. `SubPaths`=`Predicate`;
2. For i=1 to n
   (a) For each solution Sub_j in `SubPaths`
      i. For each predicate P_z in `Predicate`
         – If there are sideways passing variables from Sub_j to P_z, then add Sub= Sub_j,P_z to `NewSubPaths`
   (b) Remove from `NewSubPaths` any subpath Sub_k iff there is another subpath Sub_l in `NewSubPaths`, such that, Sub_l and Sub_k are **equivalent**, and Sub_l is **better** than Sub_k.
   (c) `SubPaths`=`NewSubPaths`
   (d) Reset `NewSubPaths`
3. Return the path in `SubPaths` with lowest cost.

Fig. 2. Query Optimization Algorithm

with 1GB RAM; experiments on real-world ontologies were executed on a Sun-Fire V440 (1281MHz) with 4GB RAM. Our system was implemented in SWI-Prolog 5.6.1.

We have studied three real-world ontologies: Travel [19], EHR_RM [21], and Galen [14].

Our cost metrics are the number of intermediate facts for synthetic and real-world ontologies, and the evaluation time for real-world ontologies. In our experiments, the sampling parameters d (the error), p (the confidence level), and k (the size of the sample for the first stage) were set to 0.2, 0.7 and 7, respectively. Also, these experiments only considered the Nested-Loop Join evaluation strategy.

Our study consisted of the following:

– *Cost Model Predictive Capability*: In Figure 3a, we report the correlation among the estimated values and the actual cost for synthetic ontologies. Synthetic ontologies were randomly generated following a uniform distribution. We generated ten ontology documents and three chain and star queries with three subgoals for each ontology; the cost of each ordering was estimated with our cost model, and each ordering was then evaluated against the ontology; this gives us a total of six hundred queries. The correlation is 0.92.
  In Figure 3b, we report the same correlation for the real-world ontology Galen. Correlation values are 0.86 for Travel, 0.54 for EHR_RM, and 0.62 for Galen.
– *Cost improvements*: We also conducted experiments to study cost improvement using the optimizer. For each query, we evaluated all its orderings, then we ran the optimizer and evaluated the optimized query. Figure 3c reports the ratio of the cost of the optimal ordering to the cost of the worst ordering, $\frac{costOptimalOrdering}{costWorstOrdering}$, for queries against synthetic ontologies. In Figure 3d, we report this metric for Galen. Both in synthetic and real-world ontologies, this ratio is less than 10% for most of the queries. We also computed the proportion of the optimal ordering cost with respect to the median ordering

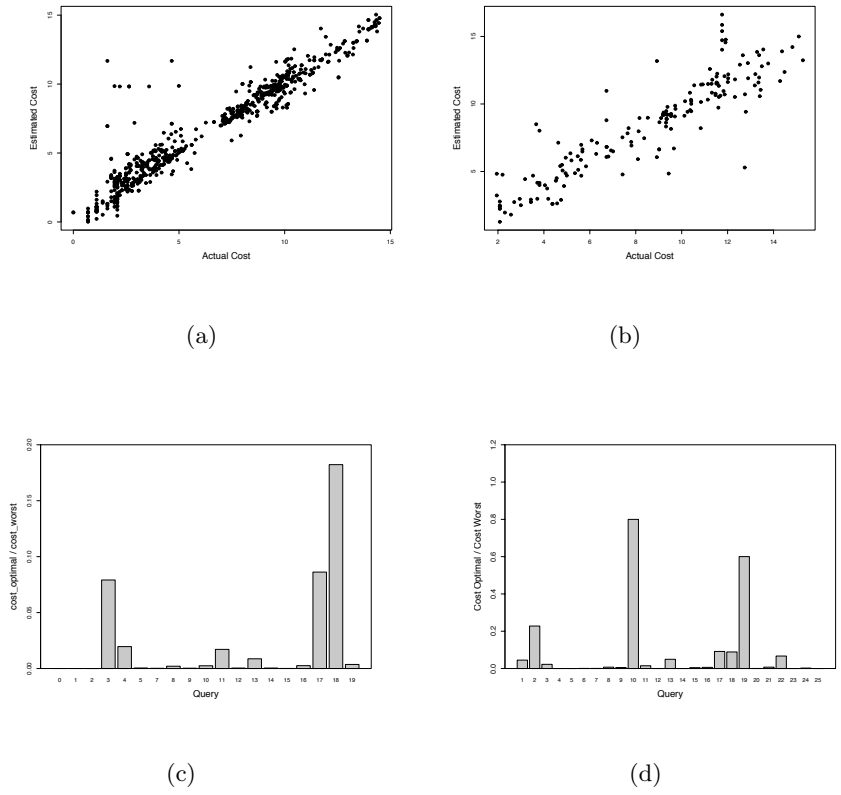


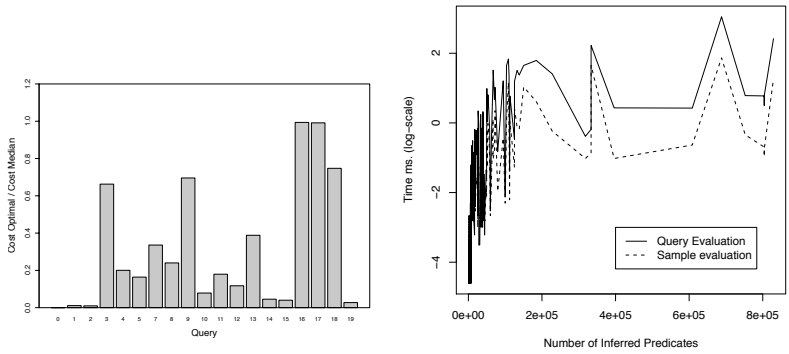
Fig. 3. (a) Correlation of estimated cost to actual cost (log. scale) - Synt. ontologies; (b) Correlation of estimated cost to actual cost (log. scale) - GALEN; (c) #Pred. optimal ordering vs. #Pred. worst ordering - Synt. Ontologies; (d) #Pred. optimal ordering vs. #Pred. worst ordering - GALEN

cost. The results for synthetic ontologies show that the optimal ordering cost is less than 40% of the median for fifteen of twenty queries; this result can be observed in Figure 4a.

Correlation results show that estimates produced by our cost model are quite accurate. The lower correlation results for the EHR_RM and Galen ontologies are related to the uniform distribution assumption of our cost model.

Additionally, the results show a significant improvement in the evaluation cost for the optimized queries with respect to the worst-case and median-case query orderings. This property holds for synthetic and real-world ontologies. However, for synthetic ontologies we notice that for star-shaped queries, the difference between the median cost and the optimal cost is very small; this

indicates that the form of the query may influence the cost improvement achieved by the optimizer.



(a) (b)

Fig. 4. (a) #Pred. optimal ordering vs. #Pred. median ordering - Synt. Ontologies; (b) Sampling Conjunctions - Query Eval. time and Sample Eval. time vs. # Inf. Pred.

Finally, we would like to point out that we also studied the use of an adaptive sampling technique for the cost estimation of the conjunction of two or more predicates (instead of System R cost model). Although, the sampling technique gives a better correlation result than the combination of sampling and System R cost model, the time required to compute the cost estimation may be as large as the time needed to evaluate the query. In 4b we can observe that the time difference is marginal.

8 Conclusions and Future Work

We have developed a cost model that combines System R and adaptive sampling techniques. Adaptive sampling is used to estimate data that do not exist a priori, data related to the cardinality and cost of intensional rules in the DOB. The experimental results show that our proposed techniques produce in general a significant improvement in the evaluation cost for the optimized query.

Currently we are concluding an experimental study that considers the three evaluation strategies: Nested-Loop, Block Nested-Loop, and Hash Join; query plans now include orderings with different combinations of these evaluation operators. Initial results show correlation values among estimated and actual cost of approximately 0.8 for real-world ontologies. We also plan to apply similar optimization techniques for conjunctive queries to DL ontologies. Initially, we will

work on ABox queries extending the the techniques proposed in [20]. In a next stage, we will consider mixed TBox and ABox conjunctive queries.

References

1. S. Abiteboul, R. Hull, and V. Vianu. Foundations of databases. *Addison-Wesly Publishing Co*, 1995.
2. J. Bruijn, A. Polleres, and D.Fensel. OWL Lite- WSML working draft. DERI institute, 2004.
3. D. Calvanese, G. Di Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tailoring OWL for data intensive ontologies. In *Proc. of the Workshop on OWL: Experiences and Directions*, 2005.
4. T. Eiter, G. Ianni, R. Schindlauer, and H. Tompits. Towards efficient evaluation of hex-programs. In *11TH Nonmonotonic Reasoning Workshop*, 2006.
5. B. Grosof, I. Horrocks, R. Volz, and S. Decker. Description logic programs: Combining logic programs with description logic. In *Proc. of WWW*, 2003.
6. V. Haarslev and R. Moller. Optimization techniques for retrieving resources described in OWL/RDF documents: First results. In *ICAI: Proceedings of the sixth AAAI*. AAAI, 2004.
7. I. Horrocks and D. Turi. The OWL instance store: System description. In *CADE-20: 20th International Conference on Automated Deduction*, 2005.
8. Ul. Hustadt and B. Motik. Description logics and disjunctive datalog the story so far. In *DL 2005 - Description Logics 2005*, 2005.
9. Y. Ling and W. Sun. A supplement to sampling-based methods for query size estimation in a database system. *SIGMOD RECORD*, 1992.
10. R. Lipton and J. Naughton. Query size estimation by adaptive sampling (extended abstract). *Proceedings of ACM Sigmod*, pages 40–46, 1990.
11. R. Lipton, J. Naughton, and D. Schneider. Practical selectivity estimation through adaptive sampling. *Proceedings of ACM Sigmod*, pages 1–10, 1990.
12. D. McGuinness and F. van Harmelen. OWL web ontology language overview. *W3C Recommendation*, 2004.
13. B. Motik, R. Volz, and A. Maedche. Optimizing query answering in description logics using disjunctive deductive databases. In *KRDB*, 2003.
14. Open Clinical Organization. GALEN common reference model.
15. Z. Pan and J. Hefflin. DLDB: Extending relational databases to support semantic web queries. In *PSSS-03: Practical and Scalable Semantic Systems*, 2003.
16. R. Ramakrishnan and J. Gehrke. *Database Management Systems*. Mc Graw Hill, 2003.
17. R. Ramakrishnan and J. Ullman. A survey of research on deductive database systems. *Journal of Logic Programming*, 1993.
18. P. Selingerl, M. Astrahan, D. Chamberlin, R. Lorie, and T. Price. Access path selection in a relational database management system. *Proceedings of ACM Sigmod*, 1979.
19. M. Shell. SchemaWeb website, 2002.
20. E. Sirin and B. Parsia. Optimizations for answering conjunctive abox queries. In *DL-06: International Workshop on Description Logics*, 2006.
21. Protege staff. Protege OWL: Ontology editor for the semantic web.
22. M. Staudt, R. Soiron, C. Quix, and M. Jarke. Query optimization for repository-based applications. In *Selected Areas in Cryptography*, 1999.
23. T. Wang. Gauging ontologies and schemas by numbers. *In WWW06 Workshop Proceedings EON2006*, 2006.