Proc. 23rd Int. Workshop on Description Logics (DL2010), CEUR-WS 573, Waterloo, Canada, 2010.

# Query Algebra and Query Optimization for Concept Assertion Retrieval

Jeffrey Pound, David Toman, Grant Weddell and Jiewen Wu

Cheriton School of Computer Science, University of Waterloo, Canada {jpound, david, j55wu, gweddell}@uwaterloo.ca

Abstract. We develop a query algebra that supports efficient assertion retrieval—a natural extension of instance retrieval. The algebra is based on previously developed techniques for indexing concept descriptions. We show how relational-style query processing, including the use of secondary indices, of multiple cascaded indices, and so on, can be used to improve query performance, and also develop general conditions that enable query reformulation.

### 1 Introduction

Instance retrieval is a well known problem in which individual names from an ABox are retrieved in response to a query. The utility of a list of individual names however, has limitations in the context of end user applications. For example, displaying a list of individual identifiers may carry little useful information for a user of a DL-based information system. In this work, we focus on a generalization of the instance retrieval problem, *concept assertion retrieval*. In the concept assertion retrieval problem, a concept describing ABox individuals is retrieved in addition to the individual names. The concept is a least subsumer in a restricted language syntax specified as a parameter to the query. This parameter, a *projection description*, is used to specify the format of the returned concept description for each individual retrieved.

Concept assertion retrieval enables new possibilities for DL-based information systems as compared to tradition instance retrieval. Queries can now provide syntactically formatted concept descriptions suitable for communicating information about ABox individuals to end users. Also, concept-based ABox representations can allow efficient evaluation of queries by using tree-based search indices. In particular, query optimization may be performed in order to exploit available indices, making query evaluation efficient by avoiding general TBox reasoning in certain scenarios.

In our model, a query consists of a concept C describing individuals of interest, and a projection description Pd describing the desired information about an individual. The queries are processed over a knowledge base  $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ , where  $\mathcal{T}$  is a TBox in a chosen DL dialect, and  $\mathcal{A}$  is an ABox containing assertions on individuals of interest. An evaluation of a query produces a set of assertions of the form  $a : C_a$  such that  $\mathcal{K} \models a : (C \sqcap C_a)$  where  $C_a$  is a least subsumer in the language defined by Pd.

As a driving application for efficient concept assertion retrieval, we consider the case of a collection of web objects with DL-based semantic annotations as an ABox, in addition to a terminology encoding general axioms over the concepts used to annotate web objects. In this scenario, a web application may embed concept assertion queries in a dynamic web page. An end user supplies search values for the queries through an interface. The evaluation of the query takes place with the resulting ABox individual names and associated concept descriptions inlined in the web page.

Example 1. Consider the case of an online dealer of photography equipment. As part of a web presence, the dealer maintains (1) a knowledge base  $\mathcal{K}$  with a terminology for digital cameras and an ABox of assertions about particular cameras available for purchase through the dealer, and (2) a collection of web pages with embedded queries over this knowledge base. For example, one of the web pages might have a query Q with a query concept C of the form

 $ProductCode = "digicam" \sqcap Price < 300$ 

paired with a projection description Pd of the form

 $(Name? \sqcap \exists Supplier. (OnlineAddress? \sqcap Rating?)).$ 

(1)

Consequently, when browsing this page, a user sees in place of Q a list of inexpensive digital cameras, with each list element displaying the name of the camera together with a sublist of supplier URL addresses and ratings for that supplier.  $\Box$ 

The example illustrates how assertions computed by our query language can resemble nested relations. Note that this is beyond the scope of more general conjunctive query languages. But also note that conjunctive queries can compute joins which are not expressible in our language. However, we believe that this is not really a requirement for browsing applications such as the above which focus on finding particular information about "objects of interest".

Our contributions are as follows:

- 1. We investigate the query optimization problem for a query algebra used in concept assertion retrieval. We show how concept-based index structures can be used to efficiently evaluate queries.
- 2. We show how query plans can be composed which eliminate the need for general TBox reasoning, by making use of precomputed information stored in indices.

Subsequent sections are organized as follows. Section 2 focuses on presenting a formal definition of our concept assertion retrieval problem. In Section 3, we show how basic operations for index scanning and projection can be extended to an algebra for manipulating sets of descriptions, and consider index-based query rewriting and index selection in this framework. Section 4 shows how purely relational algebraic expressions can be derived. Our summary comments then follow in Section 5. Note that all lemmas and the main theorem are stated without proof, but that all are straightforward (but tedious) inductions on the structure of various expressions.

### 1.1 Related Work

Our notion of concept assertion retrieval derives from an earlier notion of instance retrieval by Horrocks et al. [2] and of certain answer descriptions by ourselves in which we introduced the idea of a projection description [6]. We have also incorporated earlier work on an ordering language for DL concepts, introduced in [4], that attempts to distill comparison based reasoning that happens during search. Extensions to this language have also been explored, along with some initial experimental validation of the approach [5, 3].

# 2 Definitions

We presume the DL dialect  $\mathcal{ALC}(\mathbb{S})$  whenever we mention a knowledge base  $\mathcal{K}$ , concept C, and so on, for the remainder of the paper. However, our results apply to any dialect that has the following definition of  $\mathcal{ALC}(\mathbb{S})$  as a fragment. (This requirement can be relaxed without harm: the dialect need not support concept negation.)

**Definition 1 (Description Logic** ALC(S)). Let  $\{A, A_1, \ldots\}$ ,  $\{R, R_1, \ldots\}$ ,  $\{f, g, f_1, \ldots\}$  and  $\{a, b, \ldots\}$  denote countably infinite and disjoint sets of concept names, role names, concrete features and individual names, respectively. A concept is defined by:

where k is a finite string. A constraint C is an inclusion dependency, concept assertion, or role assertion with the respective forms  $C \sqsubseteq D$ , a : C and R(a, b). A knowledge base  $\mathcal{K}$  is a finite set of constraints. We write  $\mathcal{T}$  to denote the inclusion dependencies in  $\mathcal{K}$ , called a terminology or TBox, and write  $\mathcal{A}$  to denote the assertions in  $\mathcal{K}$ , called an ABox (where  $\mathcal{K}$  is understood from context in both cases).

The semantics of  $\mathcal{ALC}(\mathbb{S})$  is defined in the standard way based on interpretations of the form  $(\triangle \uplus \mathbb{S}, \cdot^{\mathcal{I}})$  where  $\mathbb{S}$  is a totally ordered concrete domain of finite strings that serves as range of concrete features. We use standard abbreviations such as  $C \sqcup D$  for  $\neg(\neg C \sqcap \neg D)$  and  $f \leq k$  for  $(f = k) \sqcup ((f < g) \sqcap (g = k))$ . Also, given a finite set S of  $\mathcal{ALC}(\mathbb{S})$  concepts, we write  $\sqcap S$  to denote  $\top$  if S is empty and the concept  $D_1 \sqcap \cdots \sqcap D_n$  otherwise, when  $S = \{D_1, ..., D_n\}$ .

Recall from our introductory comments that a user query (C, Pd) consists of a query concept C paired with a so-called projection description Pd. The syntax for a Pd and the sublanguage of concepts in  $ALC(\mathbb{S})$  that are induced by a Pdare defined as follows.

**Definition 2 (Projection Description).** Let f, R and C be a concrete feature, role and concept, respectively. A projection description Pd is defined by the grammar:

$$Pd ::= C? \mid f? \mid Pd_1 \sqcap Pd_2 \mid \exists R.Pd \tag{2}$$

**Definition 3 (Induced Concepts).** Let Pd be a projection description. We define the sets  $\mathcal{L}_{|Pd}$  and  $\mathcal{L}_{|Pd}^{\text{TUP}}$ , the  $\mathcal{L}$  concepts generated by Pd and  $\mathcal{L}$  tuple concepts generated by Pd, respectively, as follows:

 $\mathcal{L}_{|Pd} = \{ \Box S \mid S \subseteq_{\text{fin}} \mathcal{L}_{|Pd}^{\text{TUP}} \}, and$   $\mathcal{L}_{|Pd}^{\text{TUP}} = \begin{cases} \{C, \top\} & \text{if } Pd = \text{``C?''}; \\ \{f = k \mid k \in \mathbb{S}\} \cup \{\top\} & \text{if } Pd = \text{``f?''}; \\ \{C_1 \sqcap C_2 \mid C_1 \in \mathcal{L}_{|Pd_1}^{\text{TUP}} \land C_2 \in \mathcal{L}_{|Pd_2}^{\text{TUP}} \} \text{if } Pd = \text{``Pd}_1 \sqcap Pd_2^{''}; and \\ \{\exists R.C \mid C \in \mathcal{L}_{|Pd_1}\} & \text{if } Pd = \text{``}\exists R.Pd_1^{''}. \end{cases}$ 

Thus, for a given Pd, any concept occurring in  $\mathcal{L}_{|Pd}$  satisfies a syntactic format conforming to Pd independently of any terminology  $\mathcal{T}$ . Among all possible elements of  $\mathcal{L}_{|Pd}$  are the most informative concepts for a given concept.

**Definition 4 (Least Subsuming Concepts).** Let C, S and  $\mathcal{K}$  be a concept, set of concepts and knowledge base, respectively. We write  $\lfloor S \rfloor_{\mathcal{K}}(C)$  to denote the set of concepts  $D \in S$  that are a least subsumer of C in S with respect to  $\mathcal{K}$ , that is, where  $\mathcal{K} \models C \sqsubseteq D$ , and for which there is no other concept  $D' \in S$ such that  $\mathcal{K} \models C \sqsubseteq D'$ ,  $\mathcal{K} \models D' \sqsubseteq D$  and  $\mathcal{K} \not\models D \sqsubseteq D'$ .

**Lemma 1.** Let  $\mathcal{K}$  be an  $\mathcal{ALC}(\mathbb{S})$  knowledge base and Pd a projection description. Then the following hold for any concept C:

1.  $[\mathcal{L}_{|Pd}]_{\mathcal{K}}(C)$  is non-empty; 2.  $\mathcal{K} \models C_1 \equiv C_2$ , for any  $\{C_1, C_2\} \subseteq [\mathcal{L}_{|Pd}]_{\mathcal{K}}(C)$ ; and 3.  $[[\mathcal{L}_{|Pd}]_{\mathcal{K}}(C)]_{\{\}}(\bot)$  is singleton.

Parts 1 and 2 of Lemma 1 ensure that at least one least subsuming concept exists in  $\mathcal{L}_{|Pd}$  and that they are are semantically equivalent with respect to a given  $\mathcal{K}$ . Note that some such  $\mathcal{L}$  restriction of  $\mathcal{ALC}(\mathbb{S})$  is essential to ensure part 1, e.g., that a more general fragment that simply excludes concept negation from  $\mathcal{ALC}(\mathbb{S})$  may not have this property [1]. Also note that, although  $\mathcal{L}_{|Pd}$  is infinite in general, for any fixed and finite terminology  $\mathcal{T}$  and concept C, the language  $\mathcal{L}_{|Pd}$  restricted to the symbols used in  $\mathcal{T}$  and C is necessarily finite.

Part 3 of Lemma 1 ensures that, among the least subsuming concepts in  $\mathcal{L}_{|Pd}$  with respect to  $\mathcal{K}$ , there is a unique least subsuming concept that is *the* most informative when no knowledge of  $\mathcal{K}$  is presumed. For example, let  $\mathcal{K} = \{A \sqsubseteq (f=1)\}$  and  $Pd = (A? \sqcap f?)$ . Then  $\lfloor \mathcal{L}_{|Pd} \rfloor_{\mathcal{K}}(A) = \{A \sqcap (f=1), A \sqcap \top\}$ , and  $\lfloor \{A \sqcap (f=1), A \sqcap \top\} \rfloor_{\{1\}}(\bot) = \{A \sqcap (f=1)\}$ .

To simplify notation in the remainder of the paper, we write  $\pi_{Pd,\mathcal{K}}(C)$  as shorthand for the concept  $C_1$  such that  $\lfloor \lfloor \mathcal{L}_{|Pd} \rfloor_{\mathcal{K}}(C) \rfloor_{\{\}}(\bot) = \{C_1\}$ . The formal semantics of a user query now follows.

**Definition 5 (Query Semantics).** Let  $\mathcal{K}$  be an  $\mathcal{ALC}(\mathbb{S})$  knowledge base and Q = (C, Pd) a user query over  $\mathcal{K}$ . Then Q computes the ABox

 $\{a: \pi_{Pd,\mathcal{K}} (\sqcap \{D \mid (a:D) \in \mathcal{A}\}) \mid a \text{ occurs in } \mathcal{A} \text{ and } \mathcal{K} \models a:C\}.$ (3)

This semantics ensures that concept assertion retrieval generalizes instance retrieval. In particular, an instance retrieval query C over  $\mathcal{K}$  can be formulated as query  $(C, \top ?)$  (effectively retrieving no further information about qualifying individual names in  $\mathcal{K}$ ).

In this paper and in our current implementation, we make the simplifying assumption that a knowledge base does not contain any role assertions, a condition justified in, e.g., [2]. We also assume without loss of generality that a knowledge base will have at most one concept assertion in its ABox for any individual name a. Considered together, these assumptions imply that (3) above can be equivalently formulated as

 $\{a: \pi_{Pd,\mathcal{T}}(D) \mid (a:D) \in \mathcal{A} \text{ and } \mathcal{T} \models D \sqsubseteq C\},\$ 

which suggests two key problems for computing the results of a concept assertion query: computing least subsumers in  $\mathcal{L}_{|Pd}$  for an arbitrary projection description Pd, and finding all concept assertions a : D from a potentially large set of concepts assertions, e.g., comprising an ABox, that satisfy a selection condition given by a query concept. We consider these problems in the next section.

# 3 Indices and Query Algebra

We now introduce a *query algebra* for manipulating sets of concept descriptions. Concept assertions (and, in turn, ABoxes) are therefore *encoded* as concepts by a simple protocol based on the use of the special concrete feature *Oid* that is reserved for this purpose as follows (and we assume this correspondence for the remainder of the paper):

$$(a:C) \text{ encodes as } ((Oid = "a") \sqcap C).$$
(4)

The algebra is centered around the operations for index-based selection [4] and for concept projection [6]; however, additional operators are included that allow basic boolean combinations of queries. We show how expressions in this algebra can describe a variety of *query plans* for evaluating a user query that can vary widely in the cost of their evaluation, and we outline how several standard relational-style *query optimization techniques* can be accommodated in this framework.

### 3.1 Concept Assertions and the use of Indices

The basic *leaf* operator of our algebra is an *index scan* as introduced in [6]. This assumes that all data, including the original ABox, are stored and organized with the help of so-called *concept trees* [4]. These are search trees in which nodes correspond to concepts and in which search order is defined by an *ordering description* (or Od for short): an expression conforming to the grammar "Od ::= Un  $| f : Od | D(Od_1, Od_2)$ ". Intuitively, the productions in this grammar have the respective semantics: no explicit ordering, ordering by the value of a concrete feature f, and partition by a description D. The nesting of these constructs allows, e.g., for lexicographical ordering by several concrete features, etc. (again, see [4] for further details).

**Definition 6 (Concept Index).** Let  $\mathcal{A}$  and Od be a finite set of concept assertions and an ordering description, respectively. A concept index for  $\mathcal{A}$  and Od is a concept tree with a node for each element of  $\mathcal{A}$ , encoded as a concept, that is well-formed with respect to Od. Given a knowledge base  $\mathcal{K}$ :

- The primary index P<sup>K</sup> is a concept index for A and Oid : Un;
   A secondary index S<sup>K</sup> is a concept index for the result of a user query  $(C, Oid? \sqcap Pd)$  and some Od.

In the second case we write  $S^{\mathcal{K}} := (C, Oid? \sqcap Pd) :: Od$  to specify (or declare) such a secondary index over  $\mathcal{K}$ .

The primary index for a knowledge base always exists and is organized by the names (i.e., by the Oid feature in our representation) of the individuals described by the given ABox (hence the ordering description Oid: Un). Such an index provides an efficient way to retrieve the description associated with an individual in the ABox, given the individual's name. Also, the above definition permits the existence of any number (including zero) of secondary indices, that can be organized in various ways to support user queries<sup>1</sup>. Note that secondary indices are *essential* in our approach: they enable query evaluation to avoid (or reduce) the amount of general DL reasoning during query evaluation.

Example 2 (Concept Indices for Digital Cameras). To continue with our running example: we assume three additional secondary indices, in addition to the primary index  $P^{\mathcal{CM}}$ , are available:

 $\begin{array}{l} S_1^{\mathcal{CM}} := (\top, Oid? \sqcap ProductCode?) :: ProductCode : Oid : \text{Un}, \\ S_2^{\mathcal{CM}} := (Price < 1000, Oid? \sqcap Price?) :: Price : Oid : \text{Un}, \text{ and} \\ S_3^{\mathcal{CM}} := (\top, Oid? \sqcap Name? \sqcap \exists Supplier.(OnlineAddress? \sqcap Rating?)) :: Oid : \\ \text{Un}. \end{array}$ 

The first index,  $S_1^{\mathcal{CM}}$ , enables an efficient search for individuals by the value of the feature *ProductCode*, the second by *Price* for products costing under \$1000, and the third index enables search by the individual's name, and also stores a more elaborate projection of the concept description associated with that individual in the original ABox.  $\Box$ 

#### Query Algebra 3.2

Recall that users specify concept assertion retrieval queries as pairs (C, Pd)where C is a concept that specifies a search condition and Pd is a projection description that specifies the format of the assertions in the answer to the query. To facilitate efficient evaluation of such requests we introduce a more complex query algebra to manipulate sets of concepts (usually encoding concept assertions). The algebra allows for the use of indices to speed-up search for qualifying individuals and to retrieve appropriate concepts needed to construct answer concept assertions.

<sup>&</sup>lt;sup>1</sup> Similar to relational systems, multiple specialized indices are typically defined to support queries; this is in contrast to approaches that aspire to developing an "universal" search structure(s) to represent semantic data.

**Definition 7 (Query Algebra).** The Query Algebra consists of the six operators below, called constant query, index scan, selection, projection, intersection, union, and difference, respectively. Its syntax and semantics are as follows:

(semantics)

Q ::= C	$\{C\}$
Scan <sub>X</sub> (Q)	$\{D_1 \in X \mid \exists D_2 \in Q : \mathcal{T} \models D_1 \sqsubseteq D_2\}$
$\mid \sigma_C(Q)$	$\{D \in Q \mid \mathcal{T} \models D \sqsubseteq C\}$
$\mid \pi_{Pd}(Q)$	$\{\pi_{Pd,\mathcal{T}}(D) \mid D \in Q\}$
$  Q_1 \cap Q_2$	$\{D_1 \sqcap D_2 \mid D_1 \in Q_1, D_2 \in Q_2, \mathcal{T} \not\models (D_1 \sqcap D_2) \sqsubseteq \bot\}$
$  Q_1 \cup Q_2$	$\{D_1 \sqcap D_2 \mid D_1 \in Q_1, D_2 \in Q_2, \mathcal{T} \not\models (D_1 \sqcap D_2) \sqsubseteq \bot\}$
	$\cup \{D_1 \sqcap \top \mid D_1 \in Q_1, \forall D_2 \in Q_2 : \mathcal{T} \models (D_1 \sqcap D_2) \sqsubseteq \bot\}$
	$\cup \ \{ \mid \sqcap D_2 \mid D_2 \in Q_2, \forall D_1 \in Q_1 : T \models (D_1 \sqcap D_2) \sqsubseteq \bot \}$
$  Q_1 - Q_2$	$\{D_1 \in Q_1 \mid \forall D_2 \in Q_2 : \mathcal{T} \models (D_1 \sqcap D_2) \sqsubseteq \bot\}$

where C is a concept description and X is either a primary index or secondary index. The semantics of the queries is defined the context of the primary index  $P^{\mathcal{K}}$  and zero or more secondary indices  $\{S_1^{\mathcal{K}}, \ldots, S_n^{\mathcal{K}}\}$  and with respect to a given knowledge base  $\mathcal{K}$  with a TBox  $\mathcal{T}$ .

We say that a query is pure if all occurrences of the C construct appear only in the scope (i.e., as subexpressions) of the  $SCAN_X(Q)$  operator.

We use the notation  $Q[\mathcal{T}]$  in the remainder of the paper to make the particular TBox used in the above definition of semantics explicit.

Intuitively, each of the operators maps sets of concepts to a set of concepts, with  $\operatorname{SCAN}_X(C)$  the only leaf operator that links the algebra to the underlying *concept indices*. While not mandated by our definitions, the argument Q of a given  $\operatorname{SCAN}_X(Q)$  operator is expected to be related to the Od part of the specification of the underlying concept index X to facilitate efficient index search. For example, the index  $S_1^{\mathcal{CM}}$  from Example 2 can only be efficiently searched with descriptions of the form  $(\operatorname{ProductCode} = k)$  for some string k.

### 3.3 Concept Assertion Queries as Algebraic Expressions

In this setting, a given user query (C, Pd) can always be expressed by the algebraic expression  $\pi_{(Oid? \sqcap Pd)}(\sigma_C(\text{SCAN}_{P_{\mathcal{K}}}(\top)))$  in our algebra<sup>2</sup>. However, to benefit from the performance gains made possible by secondary indices, the algebra allows a richer space of expressions:

**Lemma 2.** Let (C, Pd) be a given query. Then the expression

 $\pi_{(Oid?\sqcap Pd)}(\sigma_C((\operatorname{SCAN}_{S^{\mathcal{K}}}(C_1)\cap\cdots\cap\operatorname{SCAN}_{S^{\mathcal{K}}}(C_n))\cap(\operatorname{SCAN}_{P^{\mathcal{K}}}(\top))) \quad (5)$ 

is equivalent to the original query, provided that (i)  $S_i^{\mathcal{K}} := (D_i, (Oid? \sqcap Pd_i)) :: Od_i, (ii) \mathcal{T} \models C \sqsubseteq (D_1 \sqcap \ldots \sqcap D_n), and (iii) C_i = \pi_{(Oid? \sqcap Pd_i), \mathcal{T}}(C), for all <math>0 < i \leq n.$ 

<sup>&</sup>lt;sup>2</sup> Note the explicit request for retrieving the individual's identifier by expanding the original projection description to  $(Oid? \sqcap Pd)$ .

Conditions (i) and (ii) ensure that the combination (intersection) of the indices  $S_i^{\mathcal{K}}$  contains sufficient data to answer the original query. The last condition is necessary to supply a sufficiently general search concept to each of the indices. (Note that using the original search concept C instead would lead to loosing answers since the concept assertions stored in the secondary indices are more general than those in the ABox, in general.)

We can also check whether an index that satisfies the conditions in Lemma 2 is useful in pruning the search; it is easy to see, e.g., that indices for which  $\mathcal{T} \models \top \sqsubseteq D_i$  and  $\mathcal{T} \models \top \sqsubseteq C_i$  always return all ABox individuals and thus cannot be useful in pruning answers to the original query. In practice, the above condition can be refined to judge applicability of an index based, e.g., on *selectivity* (the fraction of individuals retrieved using the particular selection condition  $C_i$ ).

The general form of (5) can be further simplified using the analogues of relational-style query rewrites that allow the use of *secondary indices* as follows:

**Removing Redundant Selections:** The selection operation  $\sigma_C(\cdot)$  can be removed from (5) to obtain the expression

 $\pi_{(Oid; \Box Pd)}((\operatorname{SCAN}_{S_{1}^{\mathcal{K}}}(C_{1}) \cap \dots \cap \operatorname{SCAN}_{S_{n}^{\mathcal{K}}}(C_{n})) \cap (\operatorname{SCAN}_{P^{\mathcal{K}}}(\top)), \quad (6)$ 

if  $\mathcal{T} \models (C_1 \sqcap \ldots \sqcap C_n) \sqsubseteq C$ . Since the primary index  $P^{\mathcal{K}}$  is sorted by the names of the individuals (Un), the last intersection operation in the above expression can be efficiently realized by an index nested loop join.

Index-Only Query Evaluation and Simplifying Projections: The expression (6) can be further simplified if one of the secondary indices provides assertions that conform to the final projection description  $(Oid? \sqcap Pd)$ :

 $\pi_{(Oid? \sqcap Pd)}(\pi_{Oid?}((\operatorname{SCAN}_{S_{1}^{\mathcal{K}}}(C_{1}) \cap \dots \cap \operatorname{SCAN}_{S_{n-1}^{\mathcal{K}}}(C_{n-1}))) \cap (\operatorname{SCAN}_{S_{n}^{\mathcal{K}}}(C_{n})),$ (7)

assuming the projection description in the declaration of  $S_n^{\mathcal{K}}$  is the same as  $(Oid? \sqcap Pd)$ . Note that this rewriting *completely avoids* the use of the primary index.

The rewriting coupled with the ability to store and search efficiently among descriptions yields a path to defining appropriate physical data layout designs in the form of *concept indices* and in turn to *efficient plans for answering concept assertion retrieval queries*; we elaborate on this in Section 4.

*Example 3.* Recall the running example query (1). With the help of the secondary indices defined in Example 2, we can obtain the following equivalent query expression in our algebra:

$$\pi_{(Oid?\sqcap Pd)}(\pi_{Oid?}(\operatorname{SCAN}_{S_{1}^{\mathcal{CM}}}(ProductCode = \operatorname{``digicam''}) \cap \operatorname{SCAN}_{S_{2}^{\mathcal{CM}}}(Price < 300)) \cap \operatorname{SCAN}_{S_{2}^{\mathcal{CM}}}(\top)).$$
(8)

The indices  $S_1^{\mathcal{CM}}$  and  $S_2^{\mathcal{CM}}$  fully qualify the individuals needed to answer the query and can be efficiently accessed using the concepts (ProductCode = "digicam") and (Price < 300), respectively. The expression then uses the index  $S_3^{\mathcal{CM}}$  to form the concept assertions for the answer since  $S_3^{\mathcal{CM}}$  stores the (most specific) descriptions conforming to Pd.  $\Box$ 

Index only rewriting can be generalized to cases in which the final projection description Pd is *contained* in the combination of the projection descriptions

 $Pd_i$  associated with the indices  $S_i^{\mathcal{K}}$ , i.e.,  $\pi_{Pd,\mathcal{T}}(D) = \pi_{Pd,\mathcal{T}}(\pi_{(\sqcap Pd_i),\mathcal{T}}(D))$  for all qualifying  $D^{-3}$ . Similarly, general selections can in principle be replaced by boolean combinations of index scans rather than by mere index intersections (the algebra provides the union and set difference operations) and DL reasoning can be used to test for soundness of such a rewrite. This arrangement can support, e.g., horizontal partitioning of indices and other advanced data partitioning schemes.

Now observe that  $S_3^{CM}$  is organized by the ordering description Oid: Un, and therefore that the last intersection in the expression should reduce to an efficient index look-up for each qualifying individual. Here we utilize the explicit representation of the individual names in the descriptions manipulated by the algebra: the name can now be used as a *search condition* for an index.

*Example 4.* The final algebraic version of our running example thus yields the following expression:

 $\begin{aligned} & \operatorname{SCAN}_{S_{3}^{\mathcal{CM}}}(\pi_{Oid?}(\operatorname{SCAN}_{S_{2}^{\mathcal{CM}}}(\operatorname{Price} < 300 \cap \\ & \pi_{Oid?}(\operatorname{SCAN}_{S_{1}^{\mathcal{CM}}}(\operatorname{ProductCode} = ``digicam"))))) \end{aligned}$ (9)

Note that all the selections are now performed through an appropriate index scan operation, rather than by explicit set intersections.  $\Box$ 

### 4 On Purely Structural Reasoning

There are a variety of cases in which the operators in our algebra can be evaluated with simple structural subsumption testing in place of general TBox reasoning. In this section, we characterize a general condition in which this holds for various operators in a given concept assertion query Q. Recall from Definition 7 that this happens, for example, when an evaluation of Q must correspond to an evaluation of  $Q[\{ \}]$ .

We begin by introducing a notion of *typing* for queries in terms of projection descriptions, and a *normal form* for projection descriptions that suffices for characterizing the relationship between the information content of concept projections and structural subsumption testing.

**Definition 8 (Query Typing and Projection Normalization).** Let Q and Pd be a query in the concept assertion algebra and a projection description, respectively. The type of Q, written  $\alpha(Q)$ , is a set of projection descriptions defined as follows:

ſ	$( \{ \top \}$	$if Q = $ "SCAN <sub>P</sub> ( $Q_1$ )";
	$\{Pd\}$	if $Q = \operatorname{"SCAN}_{S:=(C,Pd)::Od}(Q_1)$ ";
	$\alpha(Q_1)$	$if Q = "\sigma_C(Q_1)"$ or $, "Q_1 - Q_2"$
$\alpha(Q) = \langle \langle Q \rangle \rangle$	$\{Pd\}$	$if Q = ``\pi_{Pd}(Q_1)";$
	$\{Pd_1 \sqcap Pd_2 \mid Pd_i \in \alpha(Q_i)\}$	$if Q = "Q_1 \cap Q_2";$
	$\alpha(Q_1) \cup \alpha(Q_2)$	$if Q = "Q_1 \cup Q_2";$
l	$\{C?\}$	if $Q = "C"$ otherwise.

<sup>&</sup>lt;sup>3</sup> This condition is called *projection description refinement*; the full exploration of its properties is beyond the scope of this paper.

Also, the normal form of Pd, written norm(Pd) is an exhaustive application of the following rules to any subexpression.

 $\begin{array}{ll} 1. & (f=k)? \rightsquigarrow f?, \\ 2. & (C_1 \sqcap C_2)? \rightsquigarrow (C_1? \sqcap C_2?), \ and \\ 3. & \exists R.(Pd_1 \sqcap Pd_2) \rightsquigarrow (\exists R.Pd_1) \sqcap (\exists R.Pd_2). \end{array}$ 

Note that  $\alpha(Q)$  denotes a set of projection description. This is necessary to adequately account for our union operator. Also note that norm(Pd) contains conjunctions only at the top-level and thus can be treated as a set of conjunctionfree projection descriptions with component descriptions of the form C? or f? at the end of a (possibly empty) existential role path. For example,  $\operatorname{norm}(A? \sqcap$  $\exists R.(B? \sqcap \exists S.(f=1)?))$  denotes a conjunction of the set of projection descriptions  $\{A?, \exists R.B?, \exists R.\exists S.f?\}.$ 

With query typing and projection normalization, we are now able to state our main result of the paper:

**Theorem 1.** Let  $\mathcal{K}$  and Q be a respective knowledge base and query in the concept assertion algebra. Then  $Q = Q[\{\}]$  if at least one of the following conditions hold for any subquery  $Q_1$  of Q, where op is one of  $\cap$ ,  $\cup$  or -:

- 1.  $Q_1 = {}^{\circ}C";$ 2.  $Q_1 = {}^{\circ}SCAN_S(Q_2)"$  and  $\operatorname{norm}(Pd_1) \subseteq \operatorname{norm}(Pd_2)$  for any  $Pd_1 \in \alpha(Q_2),$ where S is defined by  $(C, Pd_2) :: Od;$
- 3.  $Q_1 = \sigma_C(Q_2)$  and  $\operatorname{norm}(Pd_1) \subseteq \operatorname{norm}(Pd_2)$  for any  $Pd_2 \in \alpha(Q_2)$  and  $Pd_1 \in \alpha(C);$
- 4.  $Q_1 = "\pi_{Pd_1}(Q_2)"$  and  $\operatorname{norm}(Pd_1) \subseteq \operatorname{norm}(Pd_2)$  for any  $Pd_2 \in \alpha(Q_2)$ ; 5.  $Q_1 = "(Q_2 \text{ op } Q_3)"$  and both  $Q_2$  and  $Q_3$  are pure; and 6.  $Q_1 = "Q_2[\mathcal{K}]"$ .

To see how the theorem applies, consider the following hypothetical query and its evaluation over a knowledge base  $\mathcal{K} = \{A \subseteq B\}$  consisting of a single inclusion dependency. (1)

$(\pi_{(Oid? \sqcap B?)}(\pi_{(Oid? \sqcap A?})))$	$_{\neg B?)}(((Oid = "a") \sqcap A))^{(1)}$	$(Oid = "b") \sqcap B)^{(i)}$	$^{(2)})^{(3)})^{(4)})^{(5)}$
	$\{a:A\}$	{b:B}	
	$\{a$	A,b:B	
	$a:(A\sqcap B),b:(\top$	$\sqcap B)\}$	
	$\{a:B,b:B\}$		

The reader can confirm from our definitions that the same evaluation ensues if "[{}]" is inserted at positions (1), (2), (3) and (5) and "[ $\mathcal{T}$ ]" at position (4), that is, that general TBox reasoning is required only for the  $\pi_{(Oid? \square A? \square B?)}(\cdot)$ operator. We conclude with a more concrete example relating to our running online digital camera case.

Example 5. Theorem 1 now allows a reformulation of query (9) to the form

$$\begin{aligned} \operatorname{SCAN}_{S_3^{CM}}(\pi_{id?}(\operatorname{SCAN}_{S_2^{CM}}(\operatorname{Price} < 300 \cap \pi_{Oid?}(\operatorname{SCAN}_{S_1^{CM}}(\operatorname{ProductCode} = "digicam")))))[\{\}] \end{aligned}$$
(10)

that completely avoids TBox reasoning.  $\Box$ 

### 5 Summary and Conclusions

The framework for *concept assertion retrieval* proposed in this paper provides a basis to introducing efficient relational-style query processing to querying the semantic web data. The main cornerstones of the approach are the ability to compute *projections* of general concepts to make properties of individuals syntactically explicit, to store such assertions in efficient tree-based search structures *indices*, and to use such data structures to efficiently evaluate queries, in particular to sidestep the need for general DL reasoning at query evaluation time.

Future research can use the proposed query algebra to develop additional tools and techniques facilitating efficient query execution, for example:

- Optimization techniques that determine optimal (or nearly optimal reformulations of user queries in the algebra or its extensions; and
- Tools that allow the users to determine what indices to create for a given set of queries.

Another direction of research is whether more complex user queries, e.g., an equivalent of conjunctive queries, can be accommodated by modest extensions to the proposed framework.

A preliminary implementation of the proposed query algebra has been completed and an experimental evaluation of engineering feasibility is underway. The full source code for this implementation, along with an evaluation workload and test data is available online at http://projection-alcd.googlecode.com/.

# References

- 1. Franz Baader, Baris Sertkaya, and Anni-Yasmin Turhan. Computing the least common subsumer w.r.t. a background terminology. J. App. Logic, 5(3):392–420, 2007.
- Ian Horrocks, Lei Li, Daniele Turi, and Sean Bechhofer. The Instance Store: DL Reasoning with Large Numbers of Individuals. In Volker Haarslev and Ralf Möller, editors, *Description Logics*, volume 104 of *CEUR Workshop Proceedings*, 2004.
- 3. Jeffery Pound. Ordering, indexing, and searching semantic data: A terminology aware index structure. University of Waterloo, MMath Thesis. 2008.
- Jeffrey Pound, Lubomir Stanchev, David Toman, and Grant Weddell. On Ordering Descriptions in a Description Logic. In 20th International Workshop on Description Logics, pages 123–134. CEUR-WS vol. 250, 2007.
- Jeffrey Pound, Lubomir Stanchev, David Toman, and Grant Weddell. On Ordering and Indexing Metadata for the Semantic Web. In 21st International Workshop on Description Logics. CEUR-WS vol. 353, 2008.
- Jeffrey Pound, David Toman, Grant Weddell, and Jiewen Wu. Concept Projection in Algebras for Computing Certain Answer Descriptions. In 22nd International Workshop on Description Logics. CEUR-WS vol. 477, 2009.