Ordering, Indexing, and Searching Semantic Data: A Terminology Aware Index Structure

by

Jeffrey Pound

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2008

Abstract

Indexing data for efficient search capabilities is a core problem in many domains of computer science. As applications centered around semantic data sources become more common, the need for more sophisticated indexing and querying capabilities arises. In particular, the need to search for specific information in the presence of a terminology or ontology (i.e. a set of logic based rules that describe concepts and their relations) becomes of particular importance, as the information a user seeks may exists as an entailment of the explicit data by means of the terminology. This variant on traditional indexing and search problems forms the foundation of a range of possible technologies for semantic data.

In this work, we propose an ordering language for specifying partial orders over semantic data items modeled as descriptions in a description logic. We then show how these orderings can be used as the basis of a search tree index for processing *concept searches* in the presence of a terminology. We study in detail the properties of the orderings and the associated index structure, and also explore a relationship between ordering descriptions called *order refinement*. A sound and complete procedure for deciding refinement is given. We also empirically evaluate a prototype implementation of our index structure, validating its potential efficacy in semantic query problems.

Acknowledgements

I would like to thank my supervisor Dr. Grant Weddell for his assistance and support in developing this work. I would also like to thank my readers, Dr. David Toman and Dr. Richard Trefler, for their participation and feedback.

I would like to send a special thanks to my family for all of their support over the years, without their encouragement I would not have made it to where I am today. Thanks to all of the teachers and professors that have taught and encouraged me throughout my academic endeavours. Lastly, I would like to thank Sol for her support and patience while working towards my degrees, and all of the friends that I have met along the way for their companionship.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The problem of managing and retrieving information efficiently is paramount in any data centric application. Recent advances in the development of standards and support technologies for semantic data models have enabled new possibilities for knowledge based applications with logical reasoning support. These advances include the *Resource Description Framework* (RDF) [25], the *Web Ontology Language* (OWL) [36], the *Semantic Web Rule Language* (SWRL) [33], and the RDF query language *SPARQL* [28].

An increasing number of applications are harnessing these technologies[1], including biological data exploration applications [34, 40], semantic web agents [15], and semantic data repositories [9, 10, 18, 20, 37]. These applications often involve large complex data sources and ontologies that need to be integrated and searched. The YAGO ontology [32], dbpedia [11], and freebase [13] are some prominent examples of such semantic data sets. Also, existing problems are finding new approaches with semantic data models, such as the *semantic search* variant of the classic information retrieval problem [16].

A common factor in all of these application areas is **the need to search for specific information in the presence of a terminology** (or ontology). This complicates traditional indexing and search methods as terminologies encode general information from which new data, or properties of existing data, can be inferred. The information encoded by a system (i.e., the data instances and a terminology) is often represented with expressive logics, usually variants of *description logics* (DLs), and require reasoning to

[1] Use case studies of some of these problems and others can be found at http://www.w3.org/2001/sw/sweo/public/UseCases/.

support querying over logically entailed data.

To support efficient retrieval of semantic data, the need for more sophisticated indexing and search capabilities arises. In particular, the potential for data management systems to take advantage of a knowledge representation language with inference capabilities becomes relevant, and an improvement of the accuracy of information retrieved during search (and potentially an expansion of the expressiveness of the query language) becomes possible as a result.

For example, consider an application in which publication data is to be modeled in such a way as to allow searching in the presence of a terminology. In particular, suppose the terminology contains the following *axiom*, where *ITEM* is a primitive concept denoting a publication item, and `subject` and `price` are concrete features. The axiom states that no item with subject "C.S." can have a price over \$10.00.

$$\mathit{ITEM} \sqcap (\texttt{subject} = \texttt{"C.S."}) \sqsubseteq (\texttt{price} \leq 10.00)$$

Now consider a search for all items less than \$12.99. We know that all items with subject "C.S." will be part of the query result, even though they may *not* explicitly encode a value for the `price` attribute. Similarly, consider a search over an ordered data structure, ordered by `price`, for all items with a price of \$15.00. We can conclude, independent of whether or not a data item encodes a value for `price`, that any item with subject "C.S." will be ordered before any item with a price of \$15.00. This knowledge enables pruning during search as a result of information inferred from a terminology.

1.1 Contributions

In this work, we propose an ordering language for semantic information modeled as statements in a description logic. We then show how these orderings can be used as the basis of a tree index for searching in the presence of a terminology. This work is relevant to semantic data oriented systems, such as intelligent web agents and semantic search engines, that have the common problem of needing to search among semantic data with the possibility of inferring relevant search information from a terminology.
The contributions of this work are as follows:

- A language for specifying *ordering descriptions* which describe partial orders over concept descriptions in a description logic is proposed.

- An *order refinement* relationship is defined which characterizes a specialization relationship between a pair of ordering descriptions based on the orderings they impose over a particular space of concept descriptions. We also present a complete procedure for deciding refinement.
- An index structure based on ordering descriptions, called a *description index*, is detailed and analysed in the context of its index and search properties.
- A proof-of-concept experimental evaluation is presented to validate the potential efficacy in using descriptions indices for real word knowledge management problems.

1.2 Related Work

In the following section we survey relevant related work. To the best of our knowledge, description indices are the first terminology aware ordered index structure for semantic data, and were initially proposed by Pound et al. in [21] (with an associated technical report [22]).

1.2.1 Indexing and Ordering

Our description index incorporates the basic notion of a binary search tree, and thus has the obvious similarities with the extensive amount of research on search trees. However, there are some particular domains of tree indexing that have a stronger relationship to our proposal.

Hellerstein et al. [19] have proposed a generalized tree structure (GiST), with Aref and Ilyas [1] extending the work to efficient generalized search trees for indexing based on spatial partitioning (SP-GiST). This notion is similar to our partition ordering construct, in which indexing can be done by partitioning based on inferring concept description membership, rather than spatial relationships. While their work does not incorporate logical inference, it is possible that one could extend their notion of key consistency to incorporate our ordering formalism (which in turn rests on DL inference). This would extend both the SP-GiST structure and our work to terminology aware, disk efficient space partition indexing with unbalanced trees. Other spatial index structures, such as quad trees [26] and R-trees [17] (as well as the many R-tree variants) share the same type of similarities.

Paparizos and Jagadish [30] proposed an extended semantics for XQuery processing in XML which incorporates an order specification. This allows for reasoning about order optimization and duplicate elimination while retaining query imposed ordering or document order requirements on the results. Their formalism also allows for incomplete data items by explicitly stating the order of incomplete items as being before or after the complete items.

Stanchev and Weddell explored how DL reasoners can be used to infer redundant orderings of indices [29], and used their notion of an extended index to address the index selection problem for embedded control applications. This notion can be modeled by our ordering language and refinement relationship, and as such can be seen as a precursor to our approach.

1.2.2 Semantic Query Processing

Databases

A database centered around storing and querying descriptions of data with inference capabilities was initially proposed by Borgida et al. with the CLASSIC data model [6]. Their approach aimed for data representation with polynomial time inference capabilities, and allowed for incomplete data descriptions with an open world assumption. Our data model is similar to that of Borgida et al., in that we model data as concept descriptions with an open world assumption, however the focus of our work is a specialized index structure which was beyond the scope of the CLASSIC proposal. Our proposal is also independent of a particular DL dialect (see Section 1.4).

RDF Stores

Current work on semantic data storage and query processing is largely driven by the semantic web community. Many different approaches and systems have been developed, such as Jena [10], Sesame [9], OWLIM [20], 3Store [18], and others. The base form for data representation is RDF triples, and storage is done either using a pluggable back-end storage device [9, 20] (often a relational database [10, 18]), or by storing the triples in a native graph representation [37]. The common component in all of these systems is the method for supporting inference. Inference capabilities for semantic web engines are generally characterized by which constructs in RDFS [24] or OWL [36] are supported. OWL comes in three variants, OWL-Lite, OWL-

DL, and OWL-Full. While none of these systems support the full expresivity of OWL-DL, they do support inference in RDFS [9, 18], a fragment of OWL-Lite [37], or all of OWL-Lite [10, 20]. Entailed information is computed either by a forward chaining algorithm which preprocesses the data, expanding it by exhaustively applying the axioms in the terminology, or by a backward chaining algorithm, in which a query is rewritten using the axioms in the terminology to query the union of every possible query that may contain a result of the initial query.

In the case of forward chaining, the resulting growth of the data set can be significant, at two to six times the original data size on average for some typical examples [31]. Additionally, maintaining the entailed data set can be non-trivial if axioms in the terminology are changed or added [35]. However, with all of the data materialized, query processing can be very efficient. Backward chaining remedies the data inflation problem, but can suffer from degraded performance as the unions of many queries can be expensive to compute. Many existing systems use a combination of forward and backward chaining to try and minimize data inflation while maximizing query evaluation times. A discussion of time/space trade-offs in RDFS reasoning can be found in [31].

Our system differs from these RDF stores in a few ways. For inference support, we make use of a DL reasoner during query evaluation. The benefit is that applications can use very expressive DL languages for knowledge representation (such as OWL-DL), and still be able to process queries that take advantage of entailed information. Also, our system need not inflate the data set, and could even compress the data by removing any explicit information that is also entailed by the terminology (intuitively, a reversal of the aforementioned forward chaining expansion). The clear downside of our proposal is that potentially expensive DL reasoning tasks are being performed during query evaluation, which could have a significant impact on performance.

Despite these differences, our proposal is not necessarily a competing model for RDF or OWL stores. In fact, our description indices could conceivably be employed in an RDF or OWL store for graph matching queries such as SPARQL queries. The description index would be used for retrieving individuals satisfying some concept description as part of a larger query. For example, one could use a description index to find all bindings of a head variable in a SPARQL query (based on the description encoded in the predicate of the query), then compute values for the remaining variables in the usual way. The availability of a DL reasoner to the RDF/OWL engines could also ease the transition to query processing with data models in more expressive logics. (The Jena RDF store lists integration with external reasoners as a target goal, but currently does not have support for it.) These suggestions however, are beyond the scope of this work. Note also that, although persistence is a straightforward extension of our work, we do not address that issue and thus our proposal is not for a data store, but rather a technology that could be used in, or as the basis of, a semantic data store.

Semantic Search

Semantic search engines, such as ESTER [4], integrate semantic querying with text retrieval as a semantic variant of traditional information retrieval. Concept descriptions are extracted from natural language or keyword queries, and the results of the semantic concept search are integrated with the results from the keyword search. Currently, these systems use approaches similar to the previously discussed forward chaining expansion, in which all entailed data is materialized to allow very efficient hash indexing for search. As such, our proposed description index could play a significant role in efficient concept searching in the presence of a terminology, or in situations when forward chaining expansions are not possible or are infeasible due to the resulting data size. Description indices could also allow efficient processing of various types of range queries, such as nearest neighbor or multidimensional ranges, as part of a semantic search predicate.

1.3 Overview

The remainder of this work is organized as follows: the remaining section of this chapter discusses description logics and their role in our notion of a database. Chapter 2 formalizes our notion of an ordering description and description index, and defines a refinement relationship between ordering descriptions. In Chapter 3 a thorough analysis of ordering descriptions and description indices is given, including the properties and performance under varying assumptions about the concept descriptions being indexed and the queries being processed. Chapter 4 then presents a sound and complete procedure for deciding refinement. Chapter 5 presents our proof-of-concept experimental evaluation, and Chapter 6 concludes with a discussion of some possible extensions to ordering descriptions and future work.

1.4 Description Logics

Description logics evolved from the development of formal languages for knowledge representation, such as KL-ONE [8] and CLASSIC [7]. There are many dialects of DLs, given by the logical constructs available in the sub-language (e.g. boolean connectives such as conjunction, and relations such as roles). The general idea is to allow one to construct *concept descriptions* which are logical statements describing some conceptual class of entities in an abstract domain. The interpretation of a concept description is the set of all individuals in the domain that are described by the concept description. For example, consider the abstract domain of publication items. A concept description may describe a particular publication, with a title and price among other features, and may also relate the description of that item to other concept descriptions, such as a *hasAuthor* relation to a concept description of an author who wrote the publication item. The interpretation of this description would be a singleton set, consisting of the actual publication item (or a reference to it).

Description logics are a class of *terminological* logics, in which general statements about concepts and their relations can be made. A collection of such statements is called a *terminology*. This terminology serves as a type of schema for database applications of DLs, defining the rules of a particular system. To continue the publication example, a terminology might express that every publication item has at least one *hasAuthor* relation. This would be expressed as a *subsumption* relationship. Intuitively, this means that the set of all things which are publication items is a subset of the set of all things which have at least one author. A subsumption expression in the terminology is generally referred to as an *axiom* or *inclusion dependency*. A thorough overview of description logics can be found in [3].

1.4.1 The Description Logic $\mathcal{ALCQ}(\mathcal{D})$

We choose the description logic dialect $\mathcal{ALCQ}(\mathcal{D})$ as an illustrative example DL for this work. It should be noted however, that our results on orderings and indexing apply to any DL dialect with a linearly ordered concrete domain. Figure 1.1 summarizes the syntax and semantics for constructs in $\mathcal{ALCQ}(\mathcal{D})$, a formal definition is given below.

Definition 1 (Description Logic $\mathcal{ALCQ}(\mathcal{D})$) *Let* $\{C, C_1, \ldots\}$ *be a set of* primitive concepts*,* $\{R, S, \ldots\}$ *a set of* roles*,* $\{f, g, \ldots\}$ *a set of* concrete features*, and* $\{k, k_1, \ldots\}$ *a set of* constants*. A* concept description *is defined by the grammar:*

$$D, E \quad ::= \quad f < g \mid f < k \mid C \mid D \sqcap E \mid \neg D \mid \exists R.D \mid (\geq \ n \ R \ D).$$

An inclusion dependency *is a* subsumption *expression of the form* $D \sqsubseteq E$*. A* terminology $\mathcal{T}$ *is a finite set of inclusion dependencies.*

An interpretation $\mathcal{I}$ *is a 3-tuple* $\langle \Delta_I, \Delta_C, \cdot^{\mathcal{I}} \rangle$ *where* Δ_I *is an arbitrary abstract domain,* Δ_C *a linearly ordered concrete domain, and* $\cdot^{\mathcal{I}}$ *an interpretation function that maps:*

- *each primitive concept* C *to a set* $C^{\mathcal{I}} \subseteq \Delta_I$*,*
- *each role* R *to a relation* $R^{\mathcal{I}} \subseteq (\Delta_I \times \Delta_I)$*,*
- *each concrete feature* f *to a total function over the abstract domain* $f^{\mathcal{I}} : \Delta_I \rightarrow \Delta_C$*,*
- *each constant* k *to a constant in* Δ_C*, and*
- *the* $<$ *symbol to the binary relation for the linear order on* Δ_C*.*
- $f < k$ *to the set* $\{e \in \Delta_I : (f)^{\mathcal{I}}(e) < (k)^{\mathcal{I}}\}$
- $f < g$ *to the set* $\{e \in \Delta_I : (f)^{\mathcal{I}}(e) < (g)^{\mathcal{I}}(e)\}$
- $D_1 \sqcap D_2$ *to the set* $(D_1)^{\mathcal{I}} \cap (D_2)^{\mathcal{I}}$
- $\neg D$ *to the set* $\Delta_I \setminus (D)^{\mathcal{I}}$
- $\exists R.D$ *to the set* $\{e \in \Delta_I : (e, e') \in (R)^{\mathcal{I}}$ *and* $e' \in (D)^{\mathcal{I}}\}$
- $(\geq \ n \ R \ D)$ *to the set* $\{e \in \Delta_I : |\{e' : (e, e') \in (R)^{\mathcal{I}}$ *and* $e' \in (D)^{\mathcal{I}}\}| \geq n\}$

An interpretation $\mathcal{I}$ *satisfies an inclusion dependency* $D \sqsubseteq E$ *if* $(D)^{\mathcal{I}} \subseteq (E)^{\mathcal{I}}$*. A terminology* $\mathcal{T}$ *entails a subsumption relationship between concept descriptions* D *and* E*, written* $\mathcal{T} \models D \sqsubseteq E$*, if* $(D)^{\mathcal{I}} \subseteq (E)^{\mathcal{I}}$ *for all interpretations* $\mathcal{I}$ *that satisfy all inclusion dependencies in* $\mathcal{T}$*.*

Syntax:	Semantics (Definition of $(D)^{\mathcal{I}}$):
$D, E ::= f < k$	$\{e \in \Delta_I : (f)^{\mathcal{I}}(e) < (k)^{\mathcal{I}}\}$
$\mid\ f < g$	$\{e \in \Delta_I : (f)^{\mathcal{I}}(e) < (g)^{\mathcal{I}}(e)\}$
$\mid\ C$	$(C)^{\mathcal{I}} \subseteq \Delta_I$
$\mid\ D_1 \sqcap D_2$	$(D_1)^{\mathcal{I}} \cap (D_2)^{\mathcal{I}}$
$\mid\ \neg D$	$\Delta_I \setminus (D)^{\mathcal{I}}$
$\mid\ \exists R.D$	$\{e \in \Delta_I : (e, e') \in (R)^{\mathcal{I}} \text{ and } e' \in (D)^{\mathcal{I}}\}$
$\mid\ (\geq\ n\ R\ D)$	$\{e \in \Delta_I : \lvert\{e' : (e, e') \in (R)^{\mathcal{I}} \text{ and } e' \in (D)^{\mathcal{I}}\}\rvert \geq n\}$

Figure 1.1: Syntax and Semantics of Concept Descriptions.

For the remainder of the paper, we also use the following derived constructs:

$$\begin{aligned} D \sqcup E &\equiv \neg(\neg D \sqcap \neg E) \\ \forall R.D &\equiv \neg\exists R.\neg D \\ \top &\equiv C \sqcup \neg C \\ \bot &\equiv \neg\top \end{aligned}$$

as well as the following derived comparisons on the concrete domain:

$$\begin{aligned} f \geq g &\equiv \neg(f < g) \\ f = g &\equiv \ (f \geq g) \sqcap (g \geq f) \\ f > g &\equiv \neg(f = g) \sqcap (f \geq g) \\ f \leq g &\equiv \neg(f > g) \end{aligned}$$

Consider a concept description of a publication data item as previously discussed. We introduce the primitive concept *ITEM* to denote a publication item, use italics for role names, and typed font for concrete features. The following concept description encodes a publication item released on January 1st, 1995. The item is related to two authors with name and address information, a publisher, and also includes a title and price feature.

$$
\begin{aligned}
\big(&\mathit{ITEM} \sqcap (\texttt{title = "A Brief History of Thyme"}) \sqcap (\texttt{price = 10.99}) \\
&\sqcap \exists \mathit{hasAuthor}.\big((\texttt{name = "John Smith"}) \sqcap (\texttt{email = "john@smith.net"}) \\
&\quad\sqcap \exists \mathit{hasMailingAddress}.(\texttt{name\_of\_state = "New York"})\big) \\
&\sqcap \exists \mathit{hasAuthor}.\big((\texttt{name = "Joe Smith"}) \sqcap (\texttt{email = "joe@domain.net"}) \\
&\quad\sqcap \exists \mathit{hasMailingAddress}.(\texttt{name\_of\_state = "Alaska"})\big) \\
&\sqcap \exists \mathit{hasPublisher}.\big((\texttt{name = "Publisher House"}) \\
&\quad\sqcap \exists \mathit{hasMailingAddress}.(\texttt{name\_of\_state = "New York"})\big) \\
&\sqcap (\texttt{release\_date = 1995-01-01})\big)
\end{aligned}
$$

The sample description above encodes a small amount of data for the publication item and each concept that it is related to by a role. In general, a single concept descriptions may encode a large amount of information and have non-trivial subsumption relationships to other concept descriptions.

Now consider the rule that every publication item must have at least one author. This would be formalized as the following inclusion dependency, which states that all items must have a *hasAuthor* role relating it to something in the abstract domain.

$$\mathit{ITEM} \sqsubseteq \exists \mathit{hasAuthor}.(\top)$$

A general restriction for an arbitrary number of n authors could be captured using a numeric restriction on the role as follows:

$$\mathit{ITEM} \sqsubseteq (\geq \ n \ \mathit{hasAuthor}\ \top)$$

The example of a price constraint given in the introductory section of this chapter is also an example of an inclusion dependency. In this case the interpretation of the inclusion dependency states that the set of all things that are items having a feature value equivalence of the concrete feature `subject` to the constant "C.S", is a subset of the set of all things that have a `price` less than or equal to \$10.00.

1.4.2 Databases and Queries

In this work, our notion of a database is simply a set of concept descriptions $\mathbf{D} = \{D_1, D_2, ..., D_n\}$. For simplicity, we use syntactic equivalence as the equality comparison for set membership. The idea is that each concept description in the database denotes an entity from the abstract domain Δ_I, such as the example publication item description previously discussed.

We consider queries composed of two basic operations, selection and sort. Our selection operator consists of a concept description, the result of which is all descriptions in the database that are subsumed by the query concept description. Our sort operator consists of an ordering requirement given by the mechanisms presented in Chapter 2. We give a formal definition of a query in Section 2.3.1. As an example selection condition, for which the previously discussed publication item description would qualify as a result is given below.

$$(\mathit{ITEM} \sqcap (\texttt{price} < 12.00))$$

The query concept description finds all descriptions in the database that denote items with a `price` feature less than \$12.00. Note that the information required to deduce that a data description qualifies as a query result may be inferred from a terminology, as descriptions in the database may not explicitly encode all relevant information.

Chapter 2

Ordering Descriptions

An ordering description specifies a partial order over a set of concept descriptions in description logic. The ordering can be specified in terms of the linear order on the underlying concrete domain, or as a partitioning based on a subsumption relationship to an arbitrary concept description called a *partitioning description*. For example, an ordering of all publication items with the concrete feature `subject` equal to "C.S." ordered by `title`, followed by all other publications ordered by `release_date` could be captured as follows:

$$D(\texttt{title} : \mathrm{Un},\ \texttt{release\_date} : \mathrm{Un})$$

where D is the partitioning description ($ITEM \sqcap$ (`subject = "C.S."`)).

In this chapter we formalize our ordering language (Section 2.1). We then explore a relationship between orderings that characterizes a specialization relationship (Section 2.2). Lastly, we show how the orderings can be used as the basis of a search tree index and define the types of queries we will consider over these index structures (Section 2.3).

2.1 Ordering Description

We now formalize our language for specifying partial orders over DL concept descriptions, initially presented in [21]. Our language is extensible in the same way as a description logic, in that new ordering capabilities can be enabled by adding additional ordering constructors. We discuss a few extensions in Section 6.1. We start with a notational definition to simplify the explanation of the ordering semantics.

Notation 1 *We write D^* to denote a description obtained from D by replacing all features f by f^*, roles R by R^*, and concepts C by C^*, and extend this notation in the natural way to apply to inclusion dependencies and terminologies.*

Conceptually, our copy notation allows us to create a unique instance of a given description or terminology. This provides a simple way to, for example, compare concrete feature values while still appealing to inference from a terminology. For example, consider a terminology with the following inclusion dependency.

$$\mathit{CHEAP\_ITEM} \sqsubseteq \mathit{ITEM} \sqcap (\texttt{price} < 5.00)$$

Now consider the following two descriptions of publication items.

- $D \equiv \mathit{CHEAP\_ITEM}$
- $E \equiv \mathit{ITEM} \sqcap (\texttt{price} = 20.00)$

We can compare the above two concept descriptions on their `price` feature with use of the terminology by posing the following subsumption test.

$$(\mathcal{T} \cup \mathcal{T}^*) \models (D \sqcap E^*) \sqsubseteq (\texttt{price} < \texttt{price}^*)$$

The subsumption test holds if the set of all things which are both a D and an E^* is a subset of the set of all things that have a `price` feature less than their `price`* feature. Since $\mathcal{T}^*$, E^* and `price`* have no influence over $\mathcal{T}$, D and `price` (they contain unique labels), we know that every `price` feature that is inferred will be a consequence of $\mathcal{T}$ and D. Likewise, every `price`* feature that is inferred will be a consequence of $\mathcal{T}^*$ and E^*. Thus, the subsumption will hold since $(\texttt{price} < 5.00)$ is ordered before $\texttt{price} = 20.00$ on the underlying concrete domain.

Definition 2 (Ordering Description) *Let D be an $\mathcal{ALCQ}(\mathcal{D})$ concept description, and f a concrete feature. An* ordering description *is defined by the following grammar:*

$$Od \quad ::= \quad \text{Un} \quad | \quad f : Od \quad | \quad D(Od, Od).$$

The constructors are called the undefined ordering*,* feature value ordering*, and* partition ordering *respectively.*

For a given terminology $\mathcal{T}$ and concept descriptions D and E, we say that D is ordered before *E* by ordering description *Od* with respect to *$\mathcal{T}$, denoted $(Od)_{\mathcal{T}}(D, E)$, if $\mathcal{T} \not\models D \sqsubseteq \bot$, $\mathcal{T} \not\models E \sqsubseteq \bot$, and at least one of the following conditions holds:*

- $Od =$ "$f : Od_1$" *and* $(\mathcal{T} \cup \mathcal{T}^*) \models (D \sqcap E^*) \sqsubseteq (f < f^*)$,
- $Od =$ "$f : Od_1$", $(Od_1)_{\mathcal{T}}(D, E)$ *and* $(\mathcal{T} \cup \mathcal{T}^*) \models (D \sqcap E^*) \sqsubseteq (f = f^*)$,
- $Od =$ "$D'(Od_1, Od_2)$", $\mathcal{T} \models D \sqsubseteq D'$ *and* $\mathcal{T} \models E \sqsubseteq \neg D'$,
- $Od =$ "$D'(Od_1, Od_2)$", $(Od_1)_{\mathcal{T}}(D, E)$ *and* $\mathcal{T} \models (D \sqcup E) \sqsubseteq D'$, *or*
- $Od =$ "$D'(Od_1, Od_2)$", $(Od_2)_{\mathcal{T}}(D, E)$ *and* $\mathcal{T} \models (D \sqcup E) \sqsubseteq \neg D'$.

Two descriptions D and E are said to be incomparable with respect to an ordering *Od* and terminology *$\mathcal{T}$ if $\neg(Od)_{\mathcal{T}}(D, E)$ and $\neg(Od)_{\mathcal{T}}(E, D)$.*

We use the undefined ordering to capture situations in which no (possibly residual) ordering between descriptions is known or needed. The feature value ordering provides an endogenous ordering capability, in which concrete feature values encoded by descriptions form the basis of the ordering. The partition ordering provides an exogenous ordering capability, in which the ordering is based on a subsumption relationship to a partitioning concept, rather than the specific data values encoded by the concept description.

2.2 Order Refinement

Reasoning about the consistency of orderings has many applications, for example, in the index selection problem [29] and in query optimization [27]. In this section we explore a relationship between ordering descriptions called *order refinement*. In Chapter 4, we present a sound and complete procedure for deciding refinement.

The refinement relationship is a generalization of equality of orderings. It characterizes when the ordering imposed by one ordering description is implied by the ordering imposed by a second ordering description. However, we are not always interested in whether this implication holds over the space of all concept descriptions. In many scenarios, such as order optimization, it is sufficient to deduce that the refinement holds over a particular space of concept descriptions, the space of query results in the order optimization example. Thus, we consider refinement with respect to a concept description in the given DL, as well as a terminology.

Definition 3 (Order Refinement) *Let $\mathcal{T}$ be a terminology, D a concept description, and Od_1 and Od_2 ordering descriptions. Then Od_1* refines *Od_2* with respect to $\mathcal{T}$ and D, *written $Od_1 \prec_{\mathcal{T},D} Od_2$, if, for all concept descriptions E_1 and E_2 such that $\mathcal{T} \models (E_1 \sqcup E_2) \sqsubseteq D$:*

$$(Od_2)_{\mathcal{T}}(E_1, E_2) \ \ \textit{implies} \ \ (Od_1)_{\mathcal{T}}(E_1, E_2).$$

Od_1 is equivalent to *Od_2* with respect to $\mathcal{T}$ and D, *written $Od_1 \approx_{\mathcal{T},D} Od_2$, if $Od_1 \prec_{\mathcal{T},D} Od_2$ and $Od_2 \prec_{\mathcal{T},D} Od_1$.*

Intuitively, if a refinement relationship holds between Od_1 and Od_2, then the ordering imposed by Od_1 is equal to or more specific than the ordering imposed by Od_2 over all possible descriptions that are subsumed by the parameter concept description. The following example illustrates an application of refinement to the order optimization problem.

Example 1 *Consider an application where descriptions of publication data are to be indexed using a description tree ordered by the ordering description "$subject : date :$ Un" with respect to some terminology $\mathcal{T}$. Specifically, the tree is ordered by a major sort on the concrete feature "subject" and a minor sort of the concrete feature "date".*

Now consider a query that retrieves all of the concept descriptions of all publications in the sorted order defined by the ordering description "$subject :$ Un". We know that sorting the descriptions is unnecessary as an in-order traversal of the tree will satisfy the required ordering. In particular, the following refinement relationship holds.

$$(subject : date : \mathrm{Un}) \prec_{\mathcal{T},\top} (subject : \mathrm{Un})$$

Now consider another query for all concept descriptions subsumed by a query concept description D, such that $\mathcal{T} \models D \sqsubseteq (subject = \text{"physics"})$, and ordered again by "date". We can again avoid sorting the query result because the following refinement relationship holds with respect to D.

$$(subject : date : \mathrm{Un}) \prec_{\mathcal{T},D} (date : \mathrm{Un})$$

Example 1 shows how refinement can be used in deciding order optimization. However, the example makes some simplifying assumptions, namely that for every data description D, it can be inferred that $\mathcal{T} \models D \sqsubseteq (subject = k)$ for some value of k in Δ_C. Section 3.2 explores the assumptions that need to be made in order to guarantee various properties of description indices.

2.3 Description Index

In this section we propose constructs which allow a database consisting of a set of concept descriptions (or *data descriptions*) to be indexed, enabling efficient concept driven search capabilities. The basis of our index structure is a partially ordered binary tree, with order given by some ordering description. Below we formalize this tree structure.

Definition 4 (Description Tree) *Let D be an $\mathcal{ALCQ}(\mathcal{D})$ concept description. A* description tree *is an ordered rooted binary tree conforming to the grammar:*

$$Tr \quad ::= \quad \langle\rangle \quad | \quad \langle D, Tr, Tr\rangle.$$

The first construct denotes an empty tree, while the second construct denotes a tree with the root node labelled *by D. We write $\langle D, L, R\rangle \in Tr$ if $\langle D, L, R\rangle$ is a subtree (node) occurring in Tr, and call any tree of the form $\langle D, \langle\rangle, \langle\rangle\rangle$ a* leaf node.

Let Tr be a description tree, Od an ordering description, and $\mathcal{T}$ a terminology. Then Tr is well formed with respect to Od and $\mathcal{T}$ *if, for all $\langle D, L, R\rangle \in Tr$,*

- $\mathcal{T} \not\models D \sqsubseteq \bot$,
- $\neg(Od)_{\mathcal{T}}(D, D')$ *for all* $\langle D', L', R'\rangle \in L$, *and*
- $\neg(Od)_{\mathcal{T}}(D', D)$ *for all* $\langle D', L', R'\rangle \in R$.

When Od and $\mathcal{T}$ are clear from context, we say simply that Tr is well formed.

With respect to an ordering description Od, the conditions for a description tree to be well formed provide the invariants for insertions of new nodes. For example, when inserting a new node for description D' in the description tree $\langle D, L, R\rangle$ where $(Od)_{\mathcal{T}}(D', D)$, then a new leaf node $\langle D', \langle\rangle, \langle\rangle\rangle$ is added in subtree L.

Definition 5 (Description Index) *Let $\mathcal{T}$ be a terminology, Od an ordering description, and Tr a well formed description tree with respect to Od and $\mathcal{T}$. A* description index *is a 3-tuple $\langle Tr, Od, \mathcal{T} \rangle$.*

2.3.1 Queries

We consider two query operators over description indices. The first being a selection operator called a *concept search*, an exhaustive search for descriptions in a description index. The second is a sort operation, consisting of an ordering requirement for the result given by an ordering description.

Definition 6 (Concept Search) *Let D_Q be a concept description and $\langle Tr, Od, \mathcal{T} \rangle$ a description index. A* concept search for description D_Q in description tree *Tr is an exhaustive search for concept descriptions in Tr such that:*

- *For all descriptions D_i in the result, $\mathcal{T} \models D_i \sqsubseteq D_Q$; and*
- *For all descriptions D_i excluded from the result, $\mathcal{T} \not\models D_i \sqsubseteq D_Q$.*

Intuitively, the result of a concept search is the exhaustive set of descriptions in the data set that are subsumed by the search concept. As such, a concept search can be though of as a data retrieval problem with *certain answer semantics* given an *open-world assumption.* A query adds an ordering requirement to the concept search problem

Definition 7 (Query) *Let D_Q be a concept description, Od_Q an ordering description, and $I = \langle Tr, Od, \mathcal{T} \rangle$ a description index. A* query *is a two tuple $\langle D_Q, Od_Q \rangle$. The result of a query is the result of a concept search for D_Q in I with the following condition:*

For all D_i, E_i in the result such that D_i appears before E_i,
$$\neg (Od_Q)_{\mathcal{T}}(E_i, D_i).$$

Chapter 3

Ordering and Indexing Analysis

The performance guarantees that can be made for description indices depend on the fragment of the ordering language used to order the index, and the properties of the concept descriptions being indexed. In this chapter we explore some properties of description indices in the context of the functionality they enable for indexing. We then define restrictions that can be enforced on the indexed concept descriptions based on a notion of sufficiency, and analyse the properties and performance of description indices in this context.

3.1 Properties of Ordering Descriptions

To provide a thorough analysis of ordering descriptions and their behaviour in the presence of different classes of data descriptions, we present some properties of ordering descriptions that identify the functionality they enable for indexing purposes. This creates a framework for general discussion of orderings and their properties.

Partial Ordering

To begin, the following property states that an ordering description is irreflexive, asymmetric, and transitive. An ordering description satisfying this property would therefore define a strict partial order over concept descriptions.

Property 1 (Partial Order) *An ordering description Od has the* partial order *property if, for any terminology $\mathcal{T}$ and concept descriptions D_1, D_2 and D_3:*

1. $\neg(Od)_{\mathcal{T}}(D_1, D_1)$;
2. *If* $(Od)_{\mathcal{T}}(D_1, D_2)$, *then* $\neg(Od)_{\mathcal{T}}(D_2, D_1)$;
3. *If* $(Od)_{\mathcal{T}}(D_1, D_2)$ *and* $(Od)_{\mathcal{T}}(D_2, D_3)$; *then* $(Od)_{\mathcal{T}}(D_1, D_3)$.

Disjointness

The following Property does not have a direct impact on the performance of description indices, but can be a useful property if one wishes to extend the query capabilities to include a count aggregate (that is, a count of objects denoted by descriptions in a query result). The following property guarantees disjointness between orderable descriptions.

Property 2 (Disjointness) *An ordering description Od has the* disjointness *property if, for any terminology $\mathcal{T}$ and concept descriptions D_1 and D_2:*

$$(Od)_{\mathcal{T}}(D_1, D_2) \text{ implies } \mathcal{T} \models (D_1 \sqcap D_2) \sqsubseteq \bot.$$

Pruning

The following two properties describe an important feature of ordering descriptions that enables pruning in description indices during a concept search. While all constructs in the current ordering language support both of these properties (see Section 3.2), it is possible to add constructs to an ordering language which may satisfy only one of the pruning conditions. Thus, we present the substitution properties individually for left and right substitution. We discuss some possible extensions to the ordering language that require this distinction in Section 6.2.

Property 3 (Left Substitution) *An ordering description Od has the* left substitution *property if, for any terminology $\mathcal{T}$ and concept descriptions D_1, D_2 and D_3:*

$$\text{If } (Od)_{\mathcal{T}}(D_1, D_2),\ \mathcal{T} \models D_3 \sqsubseteq D_2 \text{ and } \mathcal{T} \not\models D_3 \sqsubseteq \bot, \text{ then } (Od)_{\mathcal{T}}(D_1, D_3).$$

Property 4 (Right Substitution) *An ordering description Od has the* right substitution *property if, for any terminology $\mathcal{T}$ and concept descriptions D_1, D_2 and D_3:*

$$\text{If } (Od)_{\mathcal{T}}(D_1, D_2),\ \mathcal{T} \models D_3 \sqsubseteq D_1 \text{ and } \mathcal{T} \not\models D_3 \sqsubseteq \bot, \text{ then } (Od)_{\mathcal{T}}(D_3, D_2).$$

3.2 Properties of Description Indices

The properties that a description index will have depend on the underlying properties that the ordering description (used to order the index) has, in combination with the ability to maintain a well formed description tree. The following section explores the properties of description indices based on the functionality enabled for indexing. We start with some observations about properties of arbitrary ordering descriptions. The following lemma states that all possible ordering descriptions will define strict partial orders over the space of possible concept descriptions.

Lemma 1 *Let Od be an arbitrary ordering description. Then Od satisfies Property 1.*

Proof: (See Appendix A.1.) □

The following lemma which extends the pruning (substitution) properties of ordering descriptions to description indices by the nature of well formed trees.

Lemma 2 (Pruning) *Let $\langle Tr, Od, \mathcal{T}\rangle$ be a description index over sufficiently descriptive descriptions, $\langle D, L, R\rangle$ a node in Tr, and E a sufficiently descriptive concept description.*

1. *If Od satisfies Property 3 then $(Od)_{\mathcal{T}}(D, E)$ implies $\mathcal{T} \not\models D' \sqsubseteq E$ for any node $\langle D', L', R'\rangle \in L$, and*

2. *If Od satisfies Property 4 then $(Od)_{\mathcal{T}}(E, D)$ implies $\mathcal{T} \not\models D' \sqsubseteq E$ for any node $\langle D', L', R'\rangle \in R$.*

Proof: (See Appendix A.2.) □

Rotations

In order to guarantee efficient search capabilities, description indices need to be able to have rotations performed to ensure a balanced tree is maintained after insertions. The following property states that both left and right tree rotations can be performed on description indices without violating the well formedness property of the tree.

Property 5 (Tree Rotation) *A description tree Tr, well formed with respect to an ordering description Od, has the* tree rotation *property if, for any terminology $\mathcal{T}$, concept descriptions D_1 and D_2, and description trees Tr_1, Tr_2, and Tr_3 that are well formed,*

$$\langle D_1, \langle D_2, Tr_1, Tr_2 \rangle, Tr_3 \rangle \textit{ is well formed if and only if } \langle D_2, Tr_1, \langle D_1, Tr_2, Tr_3 \rangle \rangle \textit{ is well formed.}$$

Order Optimization

The last property of description indices that we are interested in, order optimization, is the ability to avoid sorting a query result when the order in which the indexed descriptions are retrieved is already consistent with the order specified by the query.

Property 6 (Order Optimization) *A description tree Tr, well formed with respect to an ordering description Od, has the* order optimization *property if, for all subsequences of nodes*

$$\langle D_1, Tr'_1, Tr''_1 \rangle, \langle D_2, Tr'_2, Tr''_2 \rangle, ..., \langle D_n, Tr'_n, Tr''_n \rangle$$

given by and in-order traversal of Tr :

$$\neg(Od)_{\mathcal{T}}(D_j, D_i) \textit{ for all } 1 \leq i < j \leq n.$$

The properties of description indices that can be guaranteed may depend on the concept descriptions being indexed. To help illustrate why this is the case, consider the following examples.

Example 2 *Consider a description index $\langle Tr, f : \mathrm{Un}, \emptyset \rangle$ in which Tr is the description tree illustrated in Figure 3.1. Tr is well formed with respect to $f : \mathrm{Un}$ since each description is satisfiable (not subsumed by $\bot$) and since the*

$$\begin{tikzcd}
& \langle (f \geq 2) \sqcap (f \leq 4), \bullet, \bullet \rangle \arrow[dl] \arrow[dr] & \\
\langle (f > 3) \sqcap (f \leq 5), \langle\,\rangle, \langle\,\rangle \rangle & & \langle (f \geq 1) \sqcap (f \leq 3), \langle\,\rangle, \langle\,\rangle \rangle
\end{tikzcd}$$

Figure 3.1: A well formed description tree with respect to the ordering description "f : Un".

root node is incomparable to both of the child nodes. Now consider a concept search for $\top$*, i.e. a search that retrieves all of the concept descriptions in* Tr*. The descriptions will be retrieved in an order (by an in-order traversal) that is not consistent with* f : Un *since the right child compares* left *of the left child with respect to* f : Un.

Example 3 *Consider a description index* $\langle Tr, f : \text{Un}, \emptyset\rangle$ *in which* Tr *is the description tree illustrated in Figure 3.1. Now consider a left or right rotation which may be necessary to maintain balance after the insertion of a new node to the right or left respectively. In the case of a right rotation, the current left child becomes the root and the current right child becomes a right descendant of the new root. This produces a tree which is not well formed since the new root compares right of a node in its right subtree. A left rotation produces an analogous problem with the root being ordered to the left of a node in its left subtree.*

We see from Example 2 that a description tree does not necessarily have the order optimization property in all cases. Example 3 illustrates a similar problem with performing rotations of description trees. One way to remedy this situation is to introduce limitations on the types of concept descriptions being indexed. This can be used to ensure, for example, that exact values can be inferred for concrete features involved in feature value orderings, and that the concept descriptions are partitionable by the partition orderings.

Definition 8 (Descriptive Sufficiency) *A concept description* D *is* sufficiently descriptive with respect to ordering description Od and terminology $\mathcal{T}$*, written* $SD_{\mathcal{T}}(D, Od)$*, if at least one of the following conditions hold:*

- *$Od =$ "Un",*
- *$Od =$ "$f : Od_1$", $SD_{\mathcal{T}}(D, Od_1)$, and $\mathcal{T} \models D \sqsubseteq (f = k)$,*
- *$Od =$ "$D'(Od_1, Od_2)$", $SD_{\mathcal{T}}(D, Od_1)$, and $\mathcal{T} \models D \sqsubseteq D'$,*
- *$Od =$ "$D'(Od_1, Od_2)$", $SD_{\mathcal{T}}(D, Od_2)$, and $\mathcal{T} \models D \sqsubseteq \neg D'$,*

for some $k \in \Delta_C$. When Od and $\mathcal{T}$ are clear from context, we say simply that D is sufficiently descriptive.

With the notion of descriptive sufficiency, we can now guarantee sound rotations for description trees and order optimization.

Lemma 3 (Rotation) *Let Od be an ordering description, $\mathcal{T}$ a terminology, and D_1 and D_2 sufficiently descriptive concept descriptions. For any description trees Tr_1, Tr_2, and Tr_3 that are well formed, $\langle D_1, \langle D_2, Tr_1, Tr_2\rangle, Tr_3\rangle$ is well formed if and only if $\langle D_2, Tr_1, \langle D_1, Tr_2, Tr_3\rangle\rangle$ is well formed.*

Proof: (See Appendix A.3.) □

Lemma 4 (Order Optimization) *Let $\mathcal{T}$ be a terminology, Od an ordering description and Tr a description tree well formed with respect to Od such that for all $\langle D, Tr_1, Tr_2\rangle \in Tr$, $SD_{\mathcal{T}}(D, Od)$. Then for all subsequences of nodes $\langle D_1, Tr'_1, Tr''_1\rangle, \langle D_2, Tr'_2, Tr''_2\rangle, ..., \langle D_n, Tr'_n, Tr''_n\rangle$ given by and in-order traversal of Tr:*

$$\neg(Od)_{\mathcal{T}}(D_j, D_i) \text{ for all } 1 \leq i < j \leq n.$$

Proof: The proof of this lemma is a direct consequence of the definition of a well formed tree. Indeed, an in-order traversal will visit descriptions in an order such that a violation of the ordering given by Od would be a violation of the well formedness property which is a contradiction. □

Table 3.1 summarizes the properties of ordering descriptions as they pertain to the functionality they enable for indexing purposes. (Note that the "−" symbol represents a non-applicable field for the Un ordering constructor as it is, by definition, always $false$.) While not enforcing descriptive sufficiency allows us to index a larger class of concept descriptions, it comes at the cost of rotations and order optimization. This clearly has an effect on the performance guarantees for concept search as we can not guarantee a balanced tree. We explore this issue further in the next section.

In Absence of Descriptive Sufficiency

	Disjoint	Prune Left	Prune Right	Rotate	Order Opt.
Un	–	–	–	✓	✓
$f : Od$	✓	✓	✓	$\times$	$\times$
$D(Od_1, Od_2)$	✓	✓	✓	$\times$	$\times$

With Descriptive Sufficiency

	Disjoint	Prune Left	Prune Right	Rotate	Order Opt.
Un	–	–	–	✓	✓
$f : Od$	✓	✓	✓	✓	✓
$D(Od_1, Od_2)$	✓	✓	✓	✓	✓

Table 3.1: Properties of Description Indices

3.3 Performance of Query Evaluation

With the indexing properties of description indices established, we can now explore the guarantees that can be made for concept searches and general query evaluation, that is, the evaluation of a query of the form $\langle D, Od \rangle$ by performing a concept search for D and returning the resulting concept descriptions in an order consistent with Od. Below we characterize a class of query concept descriptions for which performance guarantees can be made.

Definition 9 (Sufficiently Selective) *A description D is* sufficiently selective for ordering description Od with respect to terminology $\mathcal{T}$, *denoted $SS_{\mathcal{T}}(D, Od)$, if at least one of the following conditions hold:*

- $\mathcal{T} \models \top \sqsubseteq D$,
- $Od =$ "$f : Od_1$", $SS_{\mathcal{T}}(D, Od_1)$ *and* $\mathcal{T} \models D \equiv (f = k)$,
- $Od =$ "$f : Od_1$" *and* $\mathcal{T} \models D \equiv (f < k)$,
- $Od =$ "$f : Od_1$" *and* $\mathcal{T} \models D \equiv \neg(f < k)$,
- $Od =$ "$f : Od_1$" *and* $\mathcal{T} \models D \equiv (\neg(f < k) \sqcap (f < k'))$,
- $Od =$ "$D'(Od_1, Od_2)$" *and* $\mathcal{T} \models D \equiv D'$,

Concept Search

Descriptive Sufficiency:	Without	With
Un	$O(n)$	$O(n)$
$f : Od$	$O(n)$	$O(k + u \cdot lg(n))$
$D(Od_1, Od_2)$	$O(n)$	$O(k + u \cdot lg(n))$

Supported Query Evaluation

Descriptive Sufficiency:	Without	With
Un	$O(n)$	$O(n)$
$f : Od$	$O(n \cdot log(n))$	$O(k + u \cdot lg(n))$
$D(Od_1, Od_2)$	$O(n \cdot log(n))$	$O(k + u \cdot lg(n))$

Table 3.2: Performance Bounds for Description Indices With and Without Descriptive Sufficiency

- *$Od =$ "$D'(Od_1, Od_2)$", $SS_{\mathcal{T}}(D, Od_1)$ and $\mathcal{T} \models D \sqsubseteq D'$,*
- *$Od =$ "$D'(Od_1, Od_2)$", $SS_{\mathcal{T}}(D, Od_2)$ and $\mathcal{T} \models D \sqsubseteq \neg D'$,*
- *$\mathcal{T} \models D \equiv (E \sqcup E')$, $SS_{\mathcal{T}}(E, Od)$ and $SS_{\mathcal{T}}(E', Od)$,*

for some constants k and k' in Δ_C and descriptions E and E'.

Intuitively, the class of sufficiently selective query concept descriptions includes disjunctive components of point and range queries on concrete features, and queries for partitions. Additionally query concept descriptions must be partitionable by occurrences of a partition ordering in the ordering description.

By pairing the refinement relationship discussed in Section 2.2 with the notion of sufficient selectivity, we can now define a wide range of queries that can be efficiently supported by a description index.

Definition 10 (Supported Query) *Let $\mathcal{T}$ be a terminology, $Q = \langle D_Q, Od_Q \rangle$ a query, and Od an ordering description. Then Q is* supported by Od with respect to $\mathcal{T}$ *if D_Q is sufficiently selective and $Od \prec_{\mathcal{T}, D_Q} Od_Q$.*

Table 3.2 summarizes the concept search and query evaluation bounds for description indices over arbitrary and sufficiently descriptive concept descriptions, where n is the number of nodes in the index, k is the size of the search/query result, u is the number of disjunctive components in the query, and lg is a base two logarithm. In the absence of descriptive sufficiency, the worst case query evaluation degrades to a full tree scan followed by a sort operation. However, in the presence of descriptive sufficiency, we can guarantee efficient search and there is no cost increase for supporting ordering operations for supported queries. The following theorem formalizes the guarantees that can be made for supported queries.

Theorem 5 *Let $\langle Tr, Od, \mathcal{T} \rangle$ be a description index over sufficiently descriptive descriptions, and Q a supported query with u disjunctive components. Then Q can be evaluated in $O(k + u \cdot lg(n))$subsumption tests of $\mathcal{ALCQ}(\mathcal{D})$, where n is the number of nodes in Tr and k is the number of descriptions in the result.*

Proof: (See Appendix A.4.) □

Chapter 4

Deciding Order Refinement

Computing refinement poses some interesting challenges, for example consider the following refinement equivalence of ordering descriptions with respect to any arbitrary query description (i.e., $\top$) and some constant k.

$$f : D(Od, Od') \approx_{\mathcal{T},\top} D(f : Od, f : Od') \quad \textit{iff} \quad \mathcal{T} \models D \equiv (f < k)$$

The interaction between partition orderings and feature value orderings takes place because the partitioning description involves the same feature as the feature value ordering. Intuitively, a linear order on the concrete feature f followed by a partition is equivalent to a partition on the feature f followed by independent linear orderings. Partition orderings can also have complex relationships with other partition orderings depending on the relationship of the partitioning descriptions. For example, consider the following refinement equivalence, again independent of the particular query description.

$$(D_2(Od_1, D_1(Od_2, Od_3))) \approx_{\mathcal{T},\top} (D_1(D_2(Od_1, Od_2), Od_3)) \quad \textit{iff} \quad \mathcal{T} \models D_2 \sqsubseteq D_1.$$

It is worthwhile to note that these types of relationships can allow rewriting of ordering descriptions to maintain balance in the ordering description itself. This could be useful for reducing the number of subsumption tests when evaluating orderings in large ordering descriptions.

4.1 Deciding Order Refinement

We now present a procedure for deciding refinement and prove its completeness. Our approach rests on the idea of enumerating all possible *partitionings*,

$$\textsc{Part}(\mathcal{T}, D, \text{“Un”}, K) = \emptyset$$

$$\begin{aligned}\textsc{Part}(\mathcal{T}, D, \text{“}f : Od_1\text{”}, K) = \; & \bigcup_{\substack{I,J \in int(K) \\ I \textbf{ before } J}} \{\langle D \sqcap f \in I, D \sqcap f \in J\rangle\} \cup \\ & \bigcup_{I \in int(K)} \textsc{Part}(\mathcal{T}, D \sqcap f \in I, Od_1, K)\end{aligned}$$

$$\begin{aligned}\textsc{Part}(\mathcal{T}, D, \text{“}D_1(Od_1, Od_2)\text{”}, K) &= \{\langle D \sqcap D_1, D \sqcap \neg D_1\rangle\} \cup \\ & \textsc{Part}(\mathcal{T}, D \sqcap D_1, Od_1, K) \;\cup\; \textsc{Part}(\mathcal{T}, D \sqcap \neg D_1, Od_2, K)\end{aligned}$$

Figure 4.1: An Ordered Partitioning Procedure

with respect to an ordering description, that any concept description could be part of by means of a subsumption relationship. In theory, there can be an infinite number of partitions, for example, by enumerating a partition for feature equivalence to each constant in the concrete domain Δ_C for a particular concrete feature. However, in practice, the number of partitions needed to characterize an ordering is bounded by the size of the (finite) terminology. We can thus enumerate a set of possible partitions, and further enumerate all possible orderings of the partitions that must hold by the definition of the ordering description. This gives us a canonical representation of the ordering which we call an *ordered partitioning*.

Because some of the partitions will be expressed as a range over a concrete feature, we introduce the following notational convention for expressing concrete feature ranges derived from a set of constants.

Notation 2 *Let K be a set of constants. Then*

$$int(K) \;::=\; \{(a,b), [a,b), (a,b], [a,b] \;\mid\; a,b \in K \cup \{-\infty, \infty\}, a \leq b\}$$

denotes the ranges defined by K. We say a range I is before *a range J if the linear order on the concrete domain of the constants in K orders all elements of the range I before all elements of the range J.*

For a range $I \in int(K)$, we use $f \in I$ to denote the concept description constraining the concrete feature f to the range I. For example, $f \in [a,b)$ denotes the concept description $(f \geq a) \sqcap (f < b)$.

Figure 4.1 defines a procedure for computing an ordered partitioning for an ordering description Od with respect to a terminology $\mathcal{T}$, concept description D, and set of constants K. The result of this procedure is a set of pairs of concept descriptions that denote an ordering over a subspace of concept descriptions subsumed by D. For each pair $\langle D, E\rangle$ in an ordered partitioning, any concept descriptions D' and E' such that $\mathcal{T} \models D' \sqsubseteq D$ and $\mathcal{T} \models E' \sqsubseteq E$ are ordered as D' before E' by the partitioning. The following lemma establishes that the order described by the ordered partitioning is equivalent to the order described by the associated ordering description with respect to the constants appearing in the given descriptions and terminology. Determining the set of constants to use is dependent on the underlying concrete domain. In the case of the rational numbers for example, it is sufficient to simply enumerate all constants appearing in the terminology, ordering descriptions, and parameter concept description. For an integer concrete domain, the neighborhoods of each constant with size proportional to the number of concrete features must also be added. We use the following notation to denote this set of constants.

Notation 3 *Let $\mathcal{T}$ be a terminology, D a concept description, and Od_1 and Od_2 ordering descriptions. Then $findK(\mathcal{T}, Od_1, Od_2, D)$ denotes the set of all constants appearing in $\mathcal{T}$, Od_1, Od_2, and D, expanded by neighborhoods on order of the number of concrete features in $\mathcal{T}$ if required by the concrete domain.*

Lemma 6 *Let $\mathcal{T}$ be a terminology, D a concept description, Od an ordering description, and D_1 and D_2 concept descriptions such that $\mathcal{T} \models D_1 \sqcup D_2 \sqsubseteq D$. Then $(Od)_{\mathcal{T}}(D_1, D_2)$ if and only if $\mathcal{T} \models D_1 \sqsubseteq E_1$ and $\mathcal{T} \models D_2 \sqsubseteq E_2$ for some $\langle E_1, E_2\rangle \in \textsc{Part}(\mathcal{T}, D, Od, findK(\mathcal{T}, Od, D \sqcup D_1 \sqcup D_2))$.*

Proof: Soundness of the above lemma follows from the fact that the partition labels themselves are ordered by Od by construction. We can then apply Property 3 and Property 4 to show that the ordering holds between D_1 and D_2.

Completeness follows from the definition of the PART function by case analysis. For Od determined by a partition ordering constructor $D(Od_1, Od_2)$, the partition by D and $\neg D$ results in a pair in the ordered partition, along with any residual orderings covering the third case of the definition. In the case of the feature value ordering constructor, possible ranges that D_1 and D_2 can be partitioned by exist, since all constants that can be form such a partitioning are included in $findK(\mathcal{T}, Od, D \sqcup D_1 \sqcup D_2)$. $\square$

With the canonical representation of the ordering given by the ordered partitioning, we can now easily deduce if an ordered partitioning is *contained* by another ordered partitioning by testing if the first ordered partitioning can be mapped into the second. The semantics for such a mapping are defined below.

Definition 11 (Mapping Function) *Let $\mathcal{T}$ be a terminology, D a concept description, Od_1 and Od_2 ordering descriptions, K a set of constants, and $m : \textsc{Part}(\mathcal{T}, D, Od_1, K) \rightarrow \textsc{Part}(\mathcal{T}, D, Od_2, K)$ a function. Then m is a* mapping function *if $m : \langle D_1, D_2 \rangle \mapsto \langle E_1, E_2 \rangle$ implies $\mathcal{T} \models D_1 \sqsubseteq E_1$ and $\mathcal{T} \models D_2 \sqsubseteq E_2$.*

The existence of a mapping function implies that every pair in an ordered partitioning obtained from some ordering description, is contained via subsumption by a pair in an ordered partitioning obtained from a second ordering description. However, the partitions that exist in an ordered partitioning depend on the constants appearing in the terminology, ordering description, and parameter concept description. Thus the existence of a mapping function only implies a refinement relationship holds for concept descriptions that can be expressed using only these constants. In order to ensure that orderings will be consistent when ordering arbitrary concept descriptions (potentially with new constants) we prove the following lemma which ensures the addition of constants to an ordered partitioning does not affect the existence of a mapping function.

Lemma 7 *Let $\mathcal{T}$ be a terminology, D a concept description, Od_1 and Od_2 ordering descriptions, and $K = findK(\mathcal{T}, Od_1, Od_2, D)$. Then, for any finite set of constants $K' \subseteq \Delta_C$, there exists a mapping function*

$$m : \textsc{Part}(\mathcal{T}, D, Od_2, K \cup K') \mapsto \textsc{Part}(\mathcal{T}, D, Od_1, K \cup K')$$

if there exists a mapping function

$$m' : \textsc{Part}(\mathcal{T}, D, Od_2, K) \mapsto \textsc{Part}(\mathcal{T}, D, Od_1, K).$$

Proof: Consider adding a single new constant k to the set of constants K. This may create new sets of pairs on both sides of the mapping by splitting ranges in existing pairs. However, as k does not play a role in inferences using $\mathcal{T}$, the original mapping function extends to the new pairs in a natural way. For a finite set of constants, the lemma follows by repeating the above construction. □

With the extension of ordered partitionings to arbitrary (finite) sets of concept descriptions without an affect on the existence of a mapping function, we can now be sure that the containment of one mapping by another is not affected by considering the orderings of a (finite) set of arbitrary concept descriptions. This allows us to formalize necessary and sufficient conditions for computing refinement.

Theorem 8 *Let $\mathcal{T}$ be a terminology, D a concept description, Od_1 and Od_2 ordering descriptions, and $K = findK(\mathcal{T}, Od_1, Od_2, D)$. Then $Od_1 \prec_{\mathcal{T},D} Od_2$ if and only if there exists a mapping function:*

$$m : \textsc{Part}(\mathcal{T}, D, Od_2, K) \mapsto \textsc{Part}(\mathcal{T}, D, Od_1, K).$$

Proof: To show soundness, assume $\neg(Od_1 \prec_{\mathcal{T},D} Od_2)$ yet a mapping function m exists. Then $\neg(Od_1 \prec_{\mathcal{T},D} Od_2)$ implies that there exists two descriptions D and E such that $(Od_2)_{\mathcal{T}}(D, E)$ and $\neg(Od_1)_{\mathcal{T}}(D, E)$. $(Od_2)_{\mathcal{T}}(D, E)$ implies that there exists an ordered partitioning containing a pair $\langle D_2, E_2 \rangle$ such that $\mathcal{T} \models D \sqsubseteq D_2$ and $\mathcal{T} \models E \sqsubseteq E_2$ by Lemma 6. Since a mapping function m exists by our assumption, there must exist a pair $\langle D', E' \rangle$ in the ordered partitioning of Od_1 such that $\mathcal{T} \models D_2 \sqsubseteq D'$ and $\mathcal{T} \models E_2 \sqsubseteq E'$. Thus $(Od_1)_{\mathcal{T}}(D', E')$ by Lemma 6. Hence $\mathcal{T} \models D \sqsubseteq D_2$ and $\mathcal{T} \models D_2 \sqsubseteq D'$, yielding $\mathcal{T} \models D \sqsubseteq D'$. Similarly, $\mathcal{T} \models E \sqsubseteq E'$. Therefore $(Od_1)_{\mathcal{T}}(D, E)$ by Property 3 and Property 4, a contradiction.

To show completeness, assume a mapping function m does not exist yet $Od_1 \prec_{\mathcal{T},D} Od_2$. Then there must be a pair in $\langle D_1, D_2 \rangle$ in the ordered partitioning of Od_2 such that for all pairs $\langle E_1, E_2 \rangle$ in the ordered partitioning of Od_1, $\mathcal{T} \not\models D_1 \sqsubseteq E_1$ or $\mathcal{T} \not\models D_2 \sqsubseteq E_2$. Thus we have $(Od_2)_{\mathcal{T}}(D_1, D_2)$ by Lemma 6, which implies $(Od_1)_{\mathcal{T}}(D_1, D_2)$ by the assumption of a refinement relationship existing. However, this means that there is a pair $\langle D'_1, D'_2 \rangle$ in the ordered partitioning of Od_1 such that $\mathcal{T} \models D_1 \sqsubseteq D'_1$ and $\mathcal{T} \models D_2 \sqsubseteq D'_2$ by Lemma 6, which implies that there exists a mapping of $\langle D_1, D_2 \rangle$ to $\langle D'_1, D'_2 \rangle$ (since the subsumptions relationships hold) which leads to a contradiction. □

Intuitively, if everything ordered by an ordering description Od_2 falls into pairs of partitions which are equivalent or more specific than the pairs of partitions obtained from another ordering description Od_1, then the ordering imposed by Od_2 is implied by the ordering imposed by Od_1. This is because any description D subsumed by a partition description E of the ordered partitioning of Od_2, will also be subsumed by something more general than E (i.e. the mapped partition from Od_1), and thus will be ordered the same way by the mapped partition. This is, by definition, a refinement relationship.

Chapter 5

Experimental Evaluation

To validate the potential efficacy of description indices, we conducted a set of proof-of-concept experiments. The goal of these experiments was to assess the capabilities of our tree-based index structure as compared to traditional tree-based index structures. The main differing components being the more expressive ordering language of our ordering descriptions for building indices, and the subsumption checking as part of ordering decisions and for query result validation. Although it is these subsumption tests that allow us to index data by inferring information from a terminology, there does not exist an alternative tree-based indexing system that can exploit logical inference in the way our system does, meaning we have no baseline measure to compare with. As such, we apply our approach to XML indexing, a simpler problem with no complex statements existing in the terminology.

As the basis of our experiments, we used the data-centric single document benchmark from the XBench XML benchmark suite [2]. This data is a synthetic document containing publication data. We map XML data to a set of concept descriptions of a chosen entity to be indexed. We then map the XQueries to a concept description and ordering description pair, such that the concept description *describes* the entities being sought in an analogous way to the XQuery, and the ordering description captures any order requirement of the XQuery. The set of concept descriptions (extracted from the XML) that are subsumed by a query concept description therefore corresponds to the set of XML entities that are the result of the associated XQuery. Figure 5.1 shows a sample XQuery and its concept description translation. The query finds all item entities released during a given time period that have either an author or publisher from New York. (Note that long XML paths

XQuery

```
for $item in /catalog/item
where ($item/author/mailing_address/name_of_state ="New York"
or $item/publisher/mailing_address/name_of_state="New York" )
and $item/release_date gt "1995-01-01"
and $item/release_date lt "2005-01-01"
return $item
```

Concept Description

$$\begin{aligned}&\mathit{ITEM} \sqcap (\texttt{release\_date > 1995-01-01}) \sqcap (\texttt{release\_date < 2005-01-01})\\ &\sqcap\ (\exists \mathit{hasAuthor}.(\exists \mathit{hasMailingAddress}.(\texttt{name\_of\_state = "New York"}))\\ &\quad \sqcup \exists \mathit{hasPublisher}.(\exists \mathit{hasMailingAddress}.(\texttt{name\_of\_state = "New York"})))\end{aligned}$$

Figure 5.1: Example XQuery and Associated Concept Description

have been shortened for illustrative purposes.) In the concept description, *ITEM* is a primitive concept denoting a publication item, roles are italicized and concrete features written in a typed font. Entities from the XML data are encoded in an analogous way. This framework allows us to compare the performance of our description indices to XQuery indexing on equivalent problems. We note however, that our design is capable of indexing data expressed in much richer logics than are needed to encode the XML example. As such, the applicability of our model extends to domains with richer data models. Conversely, our simplistic query model is not capable of handling constructive queries like joins, and we can therefore only evaluate our system on those queries that can be encoded as a concept description and ordering pair. Our query model can not express projections either, so the result of a query is always the top level entity being indexed.

Our prototype implementation of description indices uses off-the-shelf open-source software with a small java core to link all of the components. We use the FaCT++ DL reasoner [12] for subsumption testing, the DIG interface [5] for concept description representation, and the Xerces XML parsing library [39]. Communication with FaCT++ is over a self-hosted HTTP connection.

5.1 Experimental Setup

We ran the experiments on a 1.66 GHz dual-core linux based machine with 1GB of main memory. We generated data sets in three different sizes: 10MB (2,500 items), 55MB (13,750 items), and 100 MB (25,000 items). Eight of the queries from the benchmark that are expressible as concept descriptions were selected from the XBench DC/SD benchmark as well as two additional queries of our own that illustrate some advantages of our system. The first query, labeled as Q_{21} in the experiments, is supported by a description index which takes advantage of the partition orderings. The second additional query, labeled as Q_{22} and shown in Figure 5.1, is not supported by an index, but is the only query containing a disjunction. This query illustrates the differences between navigating XML trees to find values for predicate evaluation and subsumption checking in the presence of non-determinism during query evaluation. The workload along with the schema can be found in Appendix B.

Our system and the X-Hive system [38] preprocess and index the data before query processing. In both cases, we create the appropriate indices to optimize query evaluation for each query in the benchmark. (Note that there is not necessarily an index or set of indices that can aid evaluation in every case.) In situations where there was not an obvious choice of indices to maximize query performance, index selection was done based on experimentation. The Qexo [23] and Galax [14] systems are file streaming XQuery processors and consequently do not make use of preprocessing or indexing strategies. Because the the query is processed during file loading, there is no straightforward way to measure only the query processing times. As such, we report a total time and an adjusted time for these systems. The adjusted time discounts the total measured time by a constant factor representing the average file loading time for the benchmark files as determined by experimentation.

5.2 Results

Table 5.1 summarizes the query processing times for the 2,500 item data set. Note that the query labels correspond to the numbering used by the XBench benchmark, with Q_{21} and Q_{22} being our contributed queries. It is evident from these results that our system is comparable to X-Hive, our representative indexing XML query engine. On queries 6, 8, and 14, our

	Our System	X-Hive	Qexo		Galax	
	Query Time	Query Time	Total	Adj. Time	Total	Adj. Time
Q_1	7	**4**	2652	1680	4373	3401
Q_2	1164	**1006**	2009	1037	3740	2768
Q_5	**8**	9	1664	692	3591	2619
Q_6	**22**	915	2012	1040	3907	2935
Q_8	**3**	422	1668	696	3580	2608
Q_9	**2**	4	1664	692	3603	2631
Q_{12}	**2**	69	1672	700	3550	2578
Q_{14}	**7**	701	1720	748	3612	2640
Q_{21}	**439**	9332	3910	2938	9367	8395
Q_{22}	**121**	522	3160	2188	N/A	N/A

Table 5.1: Query processing run times (msec)

system outperforms the others by a significant margin. These three queries correspond to situations in which we are able to use a description index to evaluate the queries, while the other systems are forced to perform linear scans as they can not express orderings for an index to support these queries.

Query Q_2 forces all systems to perform a linear scan of the data as no indexing is possible to aid in evaluating this query. Our system performs comparably to the other systems which is a promising result since a full linear scan of the data implies a subsumption test for every data item to determine if it is a query result.

For the remaining XBench queries, Q_1, Q_5, Q_9, Q_{12}, our system performs similarly to the other systems. These queries correspond to situations in which our description index is roughly equivalent to a traditional index used by X-Hive, i.e. a simple feature value ordering. The difference being that our system still requires subsumption checking to validate that a retrieved item qualifies as a query result (note that a simple optimization could remove this subsumption check in the case where the query consists only of a feature value equivalence, however, in the general case, subsumption checking is necessary as other components of the query could disqualify a data item as a result).

The final two rows of Table 5.1 show our two contributed queries, Q_{21} and Q_{22}. Q_{21} is a query that forces a partitioning of the data with independent sorts on each partition. Since this situation can be captured by a description

Number of Items	2500	13750	25000
Our System (Q_1)	7	11	**120**
X-Hive (Q_1)	**4**	**10**	330
Qexo (Q_1)	1680	4348	6357
Galax (Q_1)	3401	34712	97095
Our System (Q_6)	**22**	**117**	**198**
X-Hive (Q_6)	915	3838	7001
Qexo (Q_6)	1040	4111	5597
Galax (Q_6)	2935	33126	94976

Table 5.2: Comparison times for all three data sets (msec)

index, in particular by utilizing a partition ordering with different residual orders, our system is able to efficiently evaluate this supported query. Q_{22} on the other hand is not supported by an index in our system or X-Hive. This query is also the only query that contains a disjunction. Thus the difference between our system and X-Hive is that X-Hive must navigate the XML tree to find the relevant values to determine if the disjunctive predicate is satisfied, while our system delegates the task to the subsumption test in FaCT++. We attribute the performance of our system on this query to the efficient reasoning procedures of FaCT++.

One rather surprising observation from Table 5.1 is that our system performs better on Q_{22} than on Q_2. Both queries require a full scan of the data (meaning n subsumption tests) but Q_{22} contains a disjunction while Q_2 is entirely conjunctive! This discrepancy in performance is likely due to the efficient order of rule application in the tableaux reasoning procedure of FaCT++. Q_{22} contains a top-level range query over a concrete feature in addition to the disjunctive existential component (see Figure 5.1). Thus, by first evaluating the simple feature value range, the reasoner only needs to consider the disjunctive component for data items which first satisfy the concrete feature range.

To evaluate scalability, we considered the four systems over all three data sets. Table 5.2 shows the results for two representative queries across the data sets. Q_1 is taken as a representative query for which both our system and the X-Hive system use an index. Q_6 is an example of a query for which our system can exploit a description index using the partition ordering, while

X-Hive does not have an index to support the query. We see that our system scales similarly to X-Hive for Q_1, this suggests that our feature value ordering and subsumption based result verification are scalable methods. In the case of Q_6, our system still scales well even with the potentially more expensive search over a partition ordering (involving more subsumption tests).

5.3 Summary

Our experiments suggest the potential efficacy of using description indices for real-world semantic data indexing applications. When applied to known XML indexing problems, our prototype system performed comparably to existing systems, and was able to efficiently support queries from a standard benchmark that existing systems could not.

While our experimental design exercises only a small part of the logical inference capabilities of our system design, the results are none-the-less promising. We see that subsumption testing of concept descriptions during query evaluation does not have a significant effect on performance, even in the presence of large data descriptions and disjunctive statements. We acknowledge that the presence of a complex terminology and more complex data descriptions may make subsumption testing more expensive for richer data sets. However, one gains the full inference capabilities on the underlying knowledge base during indexing and query processing, which can be argued as a reasonable trade-off.

Chapter 6

Conclusions

In the following chapter, we explore some possible extensions to our ordering theory, discuss some potential directions for future work, and summarize our results.

6.1 Extensions

From our experience in considering description indices for practical applications, we have found that the ability to index nested structure could be very efficacious. In our model, indexing nested structure amounts to indexing entities based on their role relations to other entities. For example, one may want to index item concepts based on properties of the authors nested in the item descriptions. In some cases, when the role can be deduced to be functional, nested indexing can be achieved with a top level index as the nested content is restricted to have a single value. However, this is not the case in general.

Ideally, one would want to independently index the nested content, and somehow preserve a mapping back to the original entities being indexed. This is analogous to how the relational model would index one table, then map matching tuples back to another table using a foreign key constraint. However, the rules that can be expressed in a terminology makes this approach highly non-trivial in our model. Instead, we consider some simple constructs which enable an exogenous and endogenous nested indexing capability, as well as a constructor which appeals directly to subsumption testing in a DL with following grammar.

$$Od \quad ::= \quad ... \ \mid \ D(Od, Od] \ \mid \ R.f : Od \ \mid \ \sqsubseteq$$

The first new construct is a weaker variant of our partition ordering. It allows a order to be defined without explicitly deducing that the concept descriptions being ordered are contained in the partition or the complement of the partition. This allows one to, for example, partition on the existence of a role relation to a particular concept. The second ordering allows explicit access to features nested under roles. (We consider a single level of nesting for simplicity, though an extension to arbitrary levels is straightforward). The last operator one may be interested in appeals directly to subsumption testing in a description logic. To capture the intended semantics, one would need to append the following alternative conditions to Definition 2.

- $Od =$ "$D_1(Od_1, Od_2]$" and $\mathcal{T} \models D \sqsubseteq D_1$ and $\mathcal{T} \not\models E \sqsubseteq D_1$,
- $Od =$ "$D_1(Od_1, Od_2]$", $(Od_1)_{\mathcal{T}}(D, E)$ and $\mathcal{T} \models (D \sqcup E) \sqsubseteq D_1$,
- $Od =$ "$D_1(Od_1, Od_2]$", $(Od_2)_{\mathcal{T}}(D, E)$, $\mathcal{T} \not\models D \sqsubseteq D_1$, and $\mathcal{T} \not\models E \sqsubseteq D_1$,
- $Od =$ "$R.f : Od_1$" and there exists k such that $\mathcal{T} \models D \sqsubseteq \exists R.(f \leq k) \sqcap \forall R.(f \leq k)$ and $\mathcal{T} \models E \sqsubseteq \forall R.(f > k)$,
- $Od =$ "$R.f : Od_1$", $(Od_1)_{\mathcal{T}}(D, E)$, and there exists k such that $\mathcal{T} \models D \sqsubseteq \forall R.(f = k)$ and $\mathcal{T} \models E \sqsubseteq \forall R.(f = k)$,
- $Od =$ "$\sqsubseteq$", $\mathcal{T} \models D \sqsubseteq E$ and $\mathcal{T} \not\models E \sqsubseteq D$.

Table 6.1 shows the properties for the extended ordering constructs without any notion of descriptive sufficiency. We see here that left pruning is not possible, though right pruning is with the weak partition ordering or the role nested ordering. Conversely, rotations and order optimization are possible without enforcing any constraints on the concept descriptions being indexed. To gain full properties of the role nested ordering, one could impose a sufficiency condition that forces the role to be functional, however, this greatly limits the use of the operator in practical indexing situations, as roles are not likely to always contain such restrictive conditions. It is also worth noting that no level of descriptive sufficiency will ever allow left pruning or disjointness in the weak partition ordering or the subsumption ordering. This is because an "overlap" in ordered descriptions is always possible.

	Disjoint	Prune Left	Prune Right	Rotate	Order Opt.
$D(Od_1, Od_2]$	×	×	✓	✓	✓
$R.f : Od$	✓	×	×	×	×
$\sqsubseteq$	×	×	✓	✓	✓

Table 6.1: Properties of Ordering Description Extensions

6.2 Future Work

Some possible avenues for future work include a full deployable implementation, which would then open possibilities in evaluating description indices under various applications scenarios, such as A-box querying, semantic search, and SPARQL querying. More extensive experiments with richer DL dialects and complex terminologies would also help evaluate the applicability and performance of description indices. A procedure for enabling general nested indexing capabilities would be very beneficial. Lastly, an extension of the ordering language (such as is suggested in Section 6.1) could enable new possibilities for indexing. The current theory, including the refinement procedure, would need to be extended to accommodate new ordering constructs.

Appendix A

Proofs

The following proofs are based on those originally found in a technical report on ordering descriptions [22]. We note that the purpose of the "*" notation described in Notation 1 is to obtain a unique copy of the description. Consequently, we allow a description to be "copied" to arbitrary levels (e.g. D^{***}). In the following proposition, we use a superscript integer to denote the number of "*" symbols appearing after a description (or terminology). The following are some simple properties regarding the consistency of using the "*" notation.

Proposition 1

1. *If* $\mathcal{T} \models D \sqsubseteq \bot$ *and* $\mathcal{T}^* \models E^* \sqsubseteq \bot$, *then* $(\mathcal{T} \cup \mathcal{T}^*) \models (D \sqcap E^*) \sqsubseteq \bot$
2. $(\mathcal{T}^i \cup \mathcal{T}^j) \models (D^i \sqcap E^j) \sqsubseteq (f^i < f^j) \equiv (\mathcal{T}^k \cup \mathcal{T}^l) \models (D^k \sqcap E^l) \sqsubseteq (f^k < f^l)$ *where* $i \neq j$ *and* $k \neq l$.

A.1 Proof of Lemma 1

Proof: The following proofs are a structural induction on the ordering description Od. Note that we use Proposition 1 to keep the "*" notation consistent.

Consider the property $(Od)_{\mathcal{T}}(D_1, D_2)$ implies $\neg(Od)_{\mathcal{T}}(D_2, D_1)$.

- Base Case: $Od =$ "Un"
  $(Od)_{\mathcal{T}}(D_1, D_2)$ is $false$ by the definition of an ordering description, thus the implication trivially holds.

- Inductive Hypothesis: Assume $(Od_0)_{\mathcal{T}}(D_1, D_2)$ implies $\neg(Od_0)_{\mathcal{T}}(D_2, D_1)$ for some arbitrary ordering description Od_0.

- Inductive Step: Consider an ordering description Od, if $\neg(Od)_{\mathcal{T}}(D_1, D_2)$, the implication trivially holds. Consider $(Od)_{\mathcal{T}}(D_1, D_2)$. There are two possible cases for the structure of Od:

  1. Case $Od =$ "$f : Od_1$"
     Since $(Od)_{\mathcal{T}}(D_1, D_2)$, there are two possible cases:
     (a) Case $(\mathcal{T} \cup \mathcal{T}^*) \models (D_1 \sqcap D_2^*) \sqsubseteq (f = f^*)$ and $(Od_1)_{\mathcal{T}}(D_1, D_2)$.
     $(Od_1)_{\mathcal{T}}(D_1, D_2)$ implies $\neg(Od)_{\mathcal{T}}(D_2, D_1)$ by the inductive hypothesis.
     (b) Case $(\mathcal{T} \cup \mathcal{T}^*) \models (D_1 \sqcap D_2^*) \sqsubseteq (f < f^*)$.
     Assume $(Od)_{\mathcal{T}}(D_2, D_1)$ for a proof by contradiction. This implies that $(\mathcal{T}^* \cup \mathcal{T}) \models (D_2^* \sqcap D_1) \sqsubseteq (f^* < f)$. Thus $(D_2^* \sqcap D_1)$ is subsumed by both $(f < f^*)$ and $(f^* < f)$ which is only possible if $\mathcal{T} \models D_1 \sqcup D_2 \sqsubseteq \bot$. However, $(Od)_{\mathcal{T}}(D_1, D_2)$ implies that $\mathcal{T} \not\models D_1 \sqsubseteq \bot$ and $\mathcal{T} \not\models D_2 \sqsubseteq \bot$ by the definition of an ordering description, which yields a contradiction. Thus the assumption $(Od)_{\mathcal{T}}(D_2, D_1)$ does not hold and consequently $(Od)_{\mathcal{T}}(D_1, D_2)$ implies $\neg(Od)_{\mathcal{T}}(D_2, D_1)$.
  2. Case $Od =$ "$D'(Od_1, Od_2)$"
     Since $(Od)_{\mathcal{T}}(D_1, D_2)$, there are three possible cases:
     (a) Case $\mathcal{T} \models (D_1 \sqcup D_2) \sqsubseteq D'$ and $(Od_1)_{\mathcal{T}}(D_1, D_2)$.
     $(Od_1)_{\mathcal{T}}(D_1, D_2)$ implies $\neg(Od_1)_{\mathcal{T}}(D_2, D_1)$ by the inductive hypothesis.
     (b) Case $\mathcal{T} \models (D_1 \sqcup D_2) \sqsubseteq \neg D'$ and $(Od_2)_{\mathcal{T}}(D_1, D_2)$.
     $(Od_2)_{\mathcal{T}}(D_1, D_2)$ implies $\neg(Od_2)_{\mathcal{T}}(D_2, D_1)$ by the inductive hypothesis.
     (c) Case $\mathcal{T} \models D_1 \sqsubseteq D'$, and $\mathcal{T} \models D_2 \sqsubseteq \neg D'$.
     Assume $(Od)_{\mathcal{T}}(D_2, D_1)$ for a proof by contradiction. $(Od)_{\mathcal{T}}(D_1, D_2)$ implies that $\mathcal{T} \not\models D_1 \sqsubseteq \bot$ and $\mathcal{T} \not\models D_2 \sqsubseteq \bot$ from the definition of an ordering description. Additionally considering the conditions for this case yields, $\mathcal{T} \not\models D_1 \sqcup D_2 \sqsubseteq D'$

and $\mathcal{T} \not\models D_1 \sqcup D_2 \sqsubseteq \neg D'$, thus it must be the case that $\mathcal{T} \models D_2 \sqsubseteq D'$, and $\mathcal{T} \models D_1 \sqsubseteq \neg D'$ because of $(Od)_{\mathcal{T}}(D_2, D_1)$. However we know that $\mathcal{T} \models D_1 \sqsubseteq D'$ and $\mathcal{T} \models D_2 \sqsubseteq \neg D'$ from the conditions for this case, which yields a contradiction. Thus the assumption $(Od)_{\mathcal{T}}(D_2, D_1)$ does not hold and consequently $(Od)_{\mathcal{T}}(D_1, D_2)$ implies $\neg(Od)_{\mathcal{T}}(D_2, D_1)$.

Consider the property $(Od)_{\mathcal{T}}(D_1, D_2)$ and $(Od)_{\mathcal{T}}(D_2, D_3)$ implies $(Od)_{\mathcal{T}}(D_1, D_3)$.

- Base Case: Od = "Un"
  $(Od)_{\mathcal{T}}(D_1, D_2)$ and $(Od)_{\mathcal{T}}(D_2, D_3)$ are $false$ by the definition of an ordering description, thus the implication trivially holds.
- Inductive Hypothesis: Assume $(Od_0)_{\mathcal{T}}(D_1, D_2)$ and $(Od_0)_{\mathcal{T}}(D_2, D_3)$ imply $(Od_0)_{\mathcal{T}}(D_1, D_3)$ for some arbitrary ordering description Od_0.
- Inductive Step: Consider an ordering description Od, if $\neg(Od)_{\mathcal{T}}(D_1, D_2)$ or $\neg(Od)_{\mathcal{T}}(D_2, D_3)$, the implication trivially holds. Consider $(Od)_{\mathcal{T}}(D_1, D_2)$ and $(Od)_{\mathcal{T}}(D_2, D_3)$. There are two possible cases for the structure of Od:

  1. Case Od = "$f : Od_1$"
     Since $(Od)_{\mathcal{T}}(D_1, D_2)$ and $(Od)_{\mathcal{T}}(D_2, D_3)$, there are four possible cases:

     (a) Case $(\mathcal{T} \cup \mathcal{T}^*) \models (D_1 \sqcap D_2^*) \sqsubseteq (f = f^*)$, $(Od_1)_{\mathcal{T}}(D_1, D_2)$, $(\mathcal{T}^* \cup \mathcal{T}^{**}) \models (D_2^* \sqcap D_3^{**}) \sqsubseteq (f^* = f^{**})$, and $(Od_1)_{\mathcal{T}}(D_2, D_3)$. $(Od_1)_{\mathcal{T}}(D_1, D_2)$ and $(Od_1)_{\mathcal{T}}(D_2, D_3)$ imply $(Od_1)_{\mathcal{T}}(D_1, D_3)$ by the inductive hypothesis. Thus, $(Od)_{\mathcal{T}}(D_1, D_2)$ and $(Od)_{\mathcal{T}}(D_2, D_3)$ imply $(Od)_{\mathcal{T}}(D_1, D_3)$.

     (b) Case $(\mathcal{T} \cup \mathcal{T}^*) \models (D_1 \sqcap D_2^*) \sqsubseteq (f = f^*)$, $(Od_1)_{\mathcal{T}}(D_1, D_2)$, and $(\mathcal{T}^* \cup \mathcal{T}^{**}) \models (D_2^* \sqcap D_3^{**}) \sqsubseteq (f^* < f^{**})$.
     Assume $\neg(Od)_{\mathcal{T}}(D_1, D_3)$ for a proof by contradiction. This implies that $(\mathcal{T} \cup \mathcal{T}^{**}) \not\models (D_1 \sqcap D_3^{**}) \sqsubseteq (f < f^{**})$. Consider $e \in (D_1 \sqcap D_2^* \sqcap D_3^{**})^{\mathcal{I}}$. Because $e \in (D_2^* \sqcap D_3^{**})^{\mathcal{I}}$, $(f^*)^{\mathcal{I}}(e) < (f^{**})^{\mathcal{I}}(e)$. Also, because $e \in (D_1 \sqcap D_2^*)^{\mathcal{I}}$, $(f)^{\mathcal{I}}(e) = (f^*)^{\mathcal{I}}(e)$, which implies that $(f)^{\mathcal{I}}(e) < (f^{**})^{\mathcal{I}}(e)$. However, because $e \in (D_1 \sqcap D_3^{**})^{\mathcal{I}}$, $(f)^{\mathcal{I}}(e) \geq (f^{**})^{\mathcal{I}}(e)$ which is a contradiction. Thus the assumption of $\neg(Od)_{\mathcal{T}}(D_1, D_3)$ fails

and consequently $(Od)_{\mathcal{T}}(D_1, D_2)$ and $(Od)_{\mathcal{T}}(D_2, D_3)$ imply $(Od)_{\mathcal{T}}(D_1, D_3)$.

(c) Case $(\mathcal{T} \cup \mathcal{T}^*) \models (D_1 \sqcap D_2^*) \sqsubseteq (f < f^*)$, $(\mathcal{T}^* \cup \mathcal{T}^{**}) \models (D_2^* \sqcap D_3^{**}) \sqsubseteq (f^* = f^{**})$, and $(Od_1)_{\mathcal{T}}(D_2, D_3)$. Assume $\neg(Od)_{\mathcal{T}}(D_1, D_3)$ for a proof by contradiction. This implies that $(\mathcal{T} \cup \mathcal{T}^{**}) \not\models (D_1 \sqcap D_3^{**}) \sqsubseteq (f < f^{**})$. Consider $e \in (D_1 \sqcap D_2^* \sqcap D_3^{**})^{\mathcal{I}}$. Because $e \in (D_2^* \sqcap D_3^{**})^{\mathcal{I}}$, $(f^*)^{\mathcal{I}}(e) = (f^{**})^{\mathcal{I}}(e)$. Also, because $e \in (D_1 \sqcap D_2^*)^{\mathcal{I}}$, $(f)^{\mathcal{I}}(e) < (f^*)^{\mathcal{I}}(e)$, which implies that $(f)^{\mathcal{I}}(e) < (f^{**})^{\mathcal{I}}(e)$. However, because $e \in (D_1 \sqcap D_3^{**})^{\mathcal{I}}$, $(f)^{\mathcal{I}}(e) \geq (f^{**})^{\mathcal{I}}(e)$ which is a contradiction. Thus the assumption of $\neg(Od)_{\mathcal{T}}(D_1, D_3)$ fails and consequently $(Od)_{\mathcal{T}}(D_1, D_2)$ and $(Od)_{\mathcal{T}}(D_2, D_3)$ imply $(Od)_{\mathcal{T}}(D_1, D_3)$.

(d) Case $(\mathcal{T} \cup \mathcal{T}^*) \models (D_1 \sqcap D_2^*) \sqsubseteq (f < f^*)$ and $(\mathcal{T}^* \cup \mathcal{T}^{**}) \models (D_2^* \sqcap D_3^{**}) \sqsubseteq (f^* < f^{**})$. Assume $\neg(Od)_{\mathcal{T}}(D_1, D_3)$ for a proof by contradiction. This implies that $(\mathcal{T} \cup \mathcal{T}^{**}) \not\models (D_1 \sqcap D_3^{**}) \sqsubseteq (f < f^{**})$. Consider $e \in (D_1 \sqcap D_2^* \sqcap D_3^{**})^{\mathcal{I}}$. Because $e \in (D_2^* \sqcap D_3^{**})^{\mathcal{I}}$, $(f^*)^{\mathcal{I}}(e) < (f^{**})^{\mathcal{I}}(e)$. Also, because $e \in (D_1 \sqcap D_2^*)^{\mathcal{I}}$, $(f)^{\mathcal{I}}(e) < (f^*)^{\mathcal{I}}(e)$, which implies that $(f)^{\mathcal{I}}(e) < (f^{**})^{\mathcal{I}}(e)$. However, because $e \in (D_1 \sqcap D_3^{**})^{\mathcal{I}}$, $(f)^{\mathcal{I}}(e) \geq (f^{**})^{\mathcal{I}}(e)$ which is a contradiction. Thus the assumption of $\neg(Od)_{\mathcal{T}}(D_1, D_3)$ fails and consequently $(Od)_{\mathcal{T}}(D_1, D_2)$ and $(Od)_{\mathcal{T}}(D_2, D_3)$ imply $(Od)_{\mathcal{T}}(D_1, D_3)$.

2. Case $Od =$ "$D'(Od_1, Od_2)$"
Since $(Od)_{\mathcal{T}}(D_1, D_2)$ and $(Od)_{\mathcal{T}}(D_2, D_3)$, there are four possible cases:

(a) Case $\mathcal{T} \models (D_1 \sqcup D_2 \sqcup D_3) \sqsubseteq D'$, $(Od_1)_{\mathcal{T}}(D_1, D_2)$, and $(Od_1)_{\mathcal{T}}(D_2, D_3)$. $(Od_1)_{\mathcal{T}}(D_1, D_2)$, and $(Od_1)_{\mathcal{T}}(D_2, D_3)$ imply $(Od_1)_{\mathcal{T}}(D_1, D_3)$ by the inductive hypothesis. Thus, $(Od)_{\mathcal{T}}(D_1, D_2)$, and $(Od)_{\mathcal{T}}(D_2, D_3)$ imply $(Od)_{\mathcal{T}}(D_1, D_3)$ since $\mathcal{T} \models D_1 \sqcup D_3 \sqsubseteq D'$.

(b) Case $\mathcal{T} \models (D_1 \sqcup D_2) \sqsubseteq D'$, $\mathcal{T} \models D_3 \sqsubseteq \neg D'$, and $(Od_1)_{\mathcal{T}}(D_1, D_2)$. $(Od)_{\mathcal{T}}(D_1, D_3)$ by the definition of an ordering description considering $\mathcal{T} \models D_1 \sqsubseteq D'$ and $\mathcal{T} \models D_3 \sqsubseteq \neg D'$. Thus, $(Od)_{\mathcal{T}}(D_1, D_2)$, and $(Od)_{\mathcal{T}}(D_2, D_3)$ imply $(Od)_{\mathcal{T}}(D_1, D_3)$.

(c) Case $\mathcal{T} \models (D_1 \sqcup D_2 \sqcup D_3) \sqsubseteq \neg D'$, $(Od_2)_{\mathcal{T}}(D_1, D_2)$, and $(Od_2)_{\mathcal{T}}(D_2, D_3)$.

$(Od_2)_{\mathcal{T}}(D_1, D_2)$, and $(Od_2)_{\mathcal{T}}(D_2, D_3)$ imply $(Od_2)_{\mathcal{T}}(D_1, D_3)$ by the inductive hypothesis. Thus, $(Od)_{\mathcal{T}}(D_1, D_2)$, and $(Od)_{\mathcal{T}}(D_2, D_3)$ imply $(Od)_{\mathcal{T}}(D_1, D_3)$ since $\mathcal{T} \models D_1 \sqcup D_3 \sqsubseteq \neg D'$.

(d) Case $\mathcal{T} \models D_1 \sqsubseteq D'$, $\mathcal{T} \models (D_2 \sqcup D_3) \sqsubseteq \neg D'$, and $(Od_2)_{\mathcal{T}}(D_2, D_3)$.
$(Od)_{\mathcal{T}}(D_1, D_3)$ by the definition of an ordering description considering $\mathcal{T} \models D_1 \sqsubseteq D'$ and $\mathcal{T} \models D_3 \sqsubseteq \neg D'$. Thus, $(Od)_{\mathcal{T}}(D_1, D_2)$, and $(Od)_{\mathcal{T}}(D_2, D_3)$ imply $(Od)_{\mathcal{T}}(D_1, D_3)$.

Consider the property $(Od)_{\mathcal{T}}(D_1, D_2)$ implies $\mathcal{T} \models (D_1 \sqcap D_2) \sqsubseteq \bot$.

- Base Case: $Od =$ "Un".
  $(Od)_{\mathcal{T}}(D_1, D_2)$ is $false$ by the definition of an ordering description, thus the implication trivially holds.

- Inductive Hypothesis: Assume $(Od_0)_{\mathcal{T}}(D_1, D_2)$ implies $\mathcal{T} \models (D_1 \sqcap D_2) \sqsubseteq \bot$ for some arbitrary ordering description Od_0.

- Inductive Step: Consider an ordering description Od, if $\neg(Od)_{\mathcal{T}}(D_1, D_2)$, the implication trivially holds. Consider $(Od)_{\mathcal{T}}(D_1, D_2)$. There are two possible cases for the structure of Od:

  1. Case $Od =$ "$f : Od_1$"
     Since $(Od)_{\mathcal{T}}(D_1, D_2)$, there are two possible cases:

     (a) Case $(\mathcal{T} \cup \mathcal{T}^*) \models (D_1 \sqcap D_2^*) \sqsubseteq (f = f^*)$ and $(Od_1)_{\mathcal{T}}(D_1, D_2)$.
     $(Od_1)_{\mathcal{T}}(D_1, D_2)$ implies $\mathcal{T} \models (D_1 \sqcap D_2) \sqsubseteq \bot$ by the inductive hypothesis.

     (b) Case $(\mathcal{T} \cup \mathcal{T}^*) \models (D_1 \sqcap D_2^*) \sqsubseteq (f < f^*)$.
     Assume $\mathcal{T} \not\models (D_1 \sqcap D_2) \sqsubseteq \bot$ for a proof by contradiction. Consider $e \in (D_1 \sqcap D_2)^{\mathcal{I}}$, because $e \in (D_2)^{\mathcal{I}}$, it implies that $e \in (D_2^*)^{\mathcal{I}}$. We can conclude that $(f)^{\mathcal{I}}(e) = (f^*)^{\mathcal{I}}(e)$ because it is also the case that $e \in (D_1)^{\mathcal{I}}$. Because $e \in (D_1 \sqcap D_2^*)^{\mathcal{I}}$, $(\mathcal{T} \cup \mathcal{T}^*) \not\models (D_1 \sqcap D_2^*) \sqsubseteq (f < f^*)$ which is a contradiction and the assumption $\mathcal{T} \not\models (D_1 \sqcap D_2) \sqsubseteq \bot$ does not hold. Therefore, $(Od)_{\mathcal{T}}(D_1, D_2)$ implies $\mathcal{T} \models (D_1 \sqcap D_2) \sqsubseteq \bot$.

  2. Case $Od =$ "$D'(Od_1, Od_2)$"
     Since $(Od)_{\mathcal{T}}(D_1, D_2)$, there are three possible cases:

(a) Case $\mathcal{T} \models (D_1 \sqcup D_2) \sqsubseteq D'$ and $(Od_1)_{\mathcal{T}}(D_1, D_2)$.
$(Od_1)_{\mathcal{T}}(D_1, D_2)$ implies $\mathcal{T} \models (D_1 \sqcap D_2) \sqsubseteq \bot$ by the inductive hypothesis.

(b) Case $\mathcal{T} \models (D_1 \sqcup D_2) \sqsubseteq \neg D'$ and $(Od_2)_{\mathcal{T}}(D_1, D_2)$.
$(Od_2)_{\mathcal{T}}(D_1, D_2)$ implies $\mathcal{T} \models (D_1 \sqcap D_2) \sqsubseteq \bot$ by the inductive hypothesis.

(c) Case $\mathcal{T} \models D_1 \sqsubseteq D'$, and $\mathcal{T} \models D_2 \sqsubseteq \neg D'$.
$\mathcal{T} \models D_1 \sqcap D_2 \sqsubseteq D'$ since $\mathcal{T} \models D_1 \sqsubseteq D'$.
$\mathcal{T} \models D_1 \sqcap D_2 \sqsubseteq \neg D'$ since $\mathcal{T} \models D_2 \sqsubseteq \neg D'$.
$\mathcal{T} \models D_1 \sqcap D_2 \sqsubseteq D' \sqcap \neg D'$, which is equivalent to $\mathcal{T} \models D_1 \sqcap D_2 \sqsubseteq \bot$. Thus $(Od)_{\mathcal{T}}(D_1, D_2)$ implies $\mathcal{T} \models (D_1 \sqcap D_2) \sqsubseteq \bot$

Consider the property $(Od)_{\mathcal{T}}(D_1, D_2)$, $\mathcal{T} \models D_3 \sqsubseteq D_1$ and $\mathcal{T} \not\models D_3 \sqsubseteq \bot$ implies $(Od)_{\mathcal{T}}(D_3, D_2)$.

- Base Case: $Od =$ "Un".
  $(Od)_{\mathcal{T}}(D_1, D_2)$ is $false$ by the definition of an ordering description, thus the implication trivially holds.

- Inductive Hypothesis: Assume $(Od_0)_{\mathcal{T}}(D_1, D_2)$, $\mathcal{T} \models D_3 \sqsubseteq D_1$ and $\mathcal{T} \not\models D_3 \sqsubseteq \bot$ implies $(Od_0)_{\mathcal{T}}(D_3, D_2)$ for some arbitrary ordering description Od_0.

- Inductive Step: Consider an ordering description Od and concept description D_3 such that $\mathcal{T} \models D_3 \sqsubseteq D_1$ and $\mathcal{T} \not\models D_3 \sqsubseteq \bot$, if $\neg(Od)_{\mathcal{T}}(D_1, D_2)$, the implication trivially holds. Consider $(Od)_{\mathcal{T}}(D_1, D_2)$. There are two possible cases for the structure of Od:

  1. Case $Od =$ "$f : Od_1$"
     Since $(Od)_{\mathcal{T}}(D_1, D_2)$, there are two possible cases:
     (a) Case $(\mathcal{T} \cup \mathcal{T}^*) \models (D_1 \sqcap D_2^*) \sqsubseteq (f = f^*)$ and $(Od_1)_{\mathcal{T}}(D_1, D_2)$.
     Because $\mathcal{T} \models D_3 \sqsubseteq D_1$ and $(\mathcal{T} \cup \mathcal{T}^*) \models (D_1 \sqcap D_2^*) \sqsubseteq (f = f^*)$, $(\mathcal{T} \cup \mathcal{T}^*) \models (D_3 \sqcap D_2^*) \sqsubseteq (f = f^*)$. Also, $(Od_1)_{\mathcal{T}}(D_1, D_2)$, $\mathcal{T} \models D_3 \sqsubseteq D_1$ and $\mathcal{T} \not\models D_3 \sqsubseteq \bot$ implies $(Od_1)_{\mathcal{T}}(D_3, D_2)$ by the inductive hypothesis. Thus, $(Od)_{\mathcal{T}}(D_3, D_2)$.
     (b) Case $(\mathcal{T} \cup \mathcal{T}^*) \models (D_1 \sqcap D_2^*) \sqsubseteq (f < f^*)$.
     Because $\mathcal{T} \models D_3 \sqsubseteq D_1$ and $(\mathcal{T} \cup \mathcal{T}^*) \models (D_1 \sqcap D_2^*) \sqsubseteq (f < f^*)$, $(\mathcal{T} \cup \mathcal{T}^*) \models (D_3 \sqcap D_2^*) \sqsubseteq (f < f^*)$. Thus $(Od)_{\mathcal{T}}(D_3, D_2)$ by the definition of an ordering description.

2. Case Od = "$D'(Od_1, Od_2)$"
   Since $(Od)_{\mathcal{T}}(D_1, D_2)$, there are three possible cases:
   (a) Case $\mathcal{T} \models (D_1 \sqcup D_2) \sqsubseteq D'$ and $(Od_1)_{\mathcal{T}}(D_1, D_2)$.
   Because $\mathcal{T} \models D_3 \sqsubseteq D_1$ and $\mathcal{T} \models D_1 \sqsubseteq D'$, $\mathcal{T} \models D_3 \sqsubseteq D'$. $(Od_1)_{\mathcal{T}}(D_1, D_2)$, $\mathcal{T} \models D_3 \sqsubseteq D_1$ and $\mathcal{T} \not\models D_3 \sqsubseteq \bot$ implies $(Od_1)_{\mathcal{T}}(D_3, D_2)$ by the inductive hypothesis. Thus, $(Od)_{\mathcal{T}}(D_3, D_2)$.
   (b) Case $\mathcal{T} \models (D_1 \sqcup D_2) \sqsubseteq \neg D'$ and $(Od_2)_{\mathcal{T}}(D_1, D_2)$.
   Because $\mathcal{T} \models D_3 \sqsubseteq D_1$ and $\mathcal{T} \models D_1 \sqsubseteq \neg D'$, $\mathcal{T} \models D_3 \sqsubseteq \neg D'$. $(Od_2)_{\mathcal{T}}(D_1, D_2)$, $\mathcal{T} \models D_3 \sqsubseteq D_1$ and $\mathcal{T} \not\models D_3 \sqsubseteq \bot$ implies $(Od_2)_{\mathcal{T}}(D_3, D_2)$ by the inductive hypothesis. Thus, $(Od)_{\mathcal{T}}(D_3, D_2)$.
   (c) Case $\mathcal{T} \models D_1 \sqsubseteq D'$, and $\mathcal{T} \models D_2 \sqsubseteq \neg D'$.
   Because $\mathcal{T} \models D_3 \sqsubseteq D_1$ and $\mathcal{T} \models D_1 \sqsubseteq D'$, $\mathcal{T} \models D_3 \sqsubseteq D'$. Since $\mathcal{T} \models D_2 \sqsubseteq \neg D'$, $(Od)_{\mathcal{T}}(D_3, D_2)$ by the definition of an ordering description.

Consider the property $(Od)_{\mathcal{T}}(D_1, D_2)$, $\mathcal{T} \models D_3 \sqsubseteq D_2$ and $\mathcal{T} \not\models D_3 \sqsubseteq \bot$ implies $(Od)_{\mathcal{T}}(D_1, D_3)$. The proof is analogous to the previous case and is omitted.

$\square$

A.2 Proof of Lemma 2

Proof: Consider the first property $(Od)_{\mathcal{T}}(E, D)$ implies $\mathcal{T} \not\models D' \sqsubseteq E$ for any node $\langle D', L', R' \rangle \in R$. Assume there exists a node $\langle D', L', R' \rangle \in R$ such that $\mathcal{T} \models D' \sqsubseteq E$ for a proof by contradiction. Using $(Od)_{\mathcal{T}}(E, D)$ with Lemma 1 yields $(Od)_{\mathcal{T}}(D', D)$ since $\mathcal{T} \models D' \sqsubseteq E$. However, because Tr is well formed by the definition of an index, and $\langle D', L', R' \rangle \in R$, we know that $\neg(Od)_{\mathcal{T}}(D', D)$ which is a contradiction.

Consider the second property $(Od)_{\mathcal{T}}(D, E)$ implies $\mathcal{T} \not\models D' \sqsubseteq E$ for any node $\langle D', L', R' \rangle \in L$. Assume there exists a node $\langle D', L', R' \rangle \in L$ such that $\mathcal{T} \models D' \sqsubseteq E$ for a proof by contradiction. Using $(Od)_{\mathcal{T}}(D, E)$ with Lemma 1 yields $(Od)_{\mathcal{T}}(D, D')$ since $\mathcal{T} \models D' \sqsubseteq E$. However, because Tr is well formed by the definition of an index, and $\langle D', L', R'\rangle \in L$, we know that $\neg(Od)_{\mathcal{T}}(D, D')$ which is a contradiction. □

A.3 Proof of Lemma 3

Proof: For notational convenience, let A denote $\langle D_1, \langle D_2, Tr_1, Tr_2\rangle, Tr_3\rangle$, and B denote $\langle D_2, Tr_1, \langle D_1, Tr_2, Tr_3\rangle\rangle$. We will first consider the forward direction of the proof, A being well formed implies B is well formed. Assume B is not well formed for a proof by contradiction. Since, Tr_1, Tr_2, and Tr_3 are well formed, there are five possible situations in which B can violate the well formedness property:

1. $\exists\langle D', L, R\rangle \in Tr_1$ such that $(Od)_{\mathcal{T}}(D_2, D')$.
   Because A is well formed, $\neg(Od)_{\mathcal{T}}(D_2, D')$ for all $\langle D', L, R\rangle \in Tr_1$ which is a contradiction. Thus, it is not the case that $\exists\langle D', L, R\rangle \in Tr_1$ such that $(Od)_{\mathcal{T}}(D_2, D')$.

2. $\exists\langle D', L, R\rangle \in Tr_2$ such that $(Od)_{\mathcal{T}}(D_1, D')$.
   Because A is well formed, $\neg(Od)_{\mathcal{T}}(D_1, D')$ for all $\langle D', L, R\rangle \in Tr_2$ which is a contradiction. Thus, it is not the case that $\exists\langle D', L, R\rangle \in Tr_2$ such that $(Od)_{\mathcal{T}}(D_1, D')$.

3. $\exists\langle D', L, R\rangle \in Tr_2$ such that $(Od)_{\mathcal{T}}(D', D_2)$.
   Because A is well formed, $\neg(Od)_{\mathcal{T}}(D', D_2)$ for all $\langle D', L, R\rangle \in Tr_2$ which is a contradiction. Thus, it is not the case that $\exists\langle D', L, R\rangle \in Tr_2$ such that $(Od)_{\mathcal{T}}(D', D_2)$.

4. $\exists\langle D', L, R\rangle \in Tr_3$ such that $(Od)_{\mathcal{T}}(D', D_1)$.
   Because A is well formed, $\neg(Od)_{\mathcal{T}}(D', D_1)$ for all $\langle D', L, R\rangle \in Tr_3$ which is a contradiction. Thus, it is not the case that $\exists\langle D', L, R\rangle \in Tr_3$ such that $(Od)_{\mathcal{T}}(D', D_1)$.

5. $\exists\langle D', L, R\rangle \in Tr_3$ such that $(Od)_{\mathcal{T}}(D', D_2)$.
   Because A is well formed, $\neg(Od)_{\mathcal{T}}(D', D_1)$ for all $\langle D', L, R\rangle \in Tr_3$ and $\neg(Od)_{\mathcal{T}}(D_1, D_2)$ which yields four possibilities:

   (a) $\neg(Od)_{\mathcal{T}}(D_2, D_1)$ and $(Od)_{\mathcal{T}}(D_1, D')$. Given that D_1 and D_2 are sufficiently descriptive and incomparable, we get $(Od)_{\mathcal{T}}(D_2, D')$ from Lemma 1 using $(Od)_{\mathcal{T}}(D_1, D')$, which is a contradiction.

(b) $\neg(Od)_{\mathcal{T}}(D_2, D_1)$ and $\neg(Od)_{\mathcal{T}}(D_1, D')$. Given that D_1 and D' are sufficiently descriptive and incomparable, we get $(Od)_{\mathcal{T}}(D_1, D_2)$ from Lemma 1 using $(Od)_{\mathcal{T}}(D', D_2)$, which is a contradiction with $\neg(Od)_{\mathcal{T}}(D_1, D_2)$ from A being well formed.

(c) $(Od)_{\mathcal{T}}(D_2, D_1)$ and $(Od)_{\mathcal{T}}(D_1, D')$. Using the transitivity property of Lemma 1 with $(Od)_{\mathcal{T}}(D', D_2)$ and $(Od)_{\mathcal{T}}(D_2, D_1)$, we get $(Od)_{\mathcal{T}}(D', D_1)$ which is a contradiction with the conditions for this case.

(d) $(Od)_{\mathcal{T}}(D_2, D_1)$ and $\neg(Od)_{\mathcal{T}}(D_1, D')$. Given that D_1 and D' are sufficiently descriptive and incomparable, we get $(Od)_{\mathcal{T}}(D_2, D')$ from Lemma 1 using $(Od)_{\mathcal{T}}(D_2, D_1)$, which is a contradiction with the initial assumption.

Thus, it is not the case that $\exists\langle D', L, R\rangle \in Tr_3$ such that $(Od)_{\mathcal{T}}(D', D_2)$.

From this we can conclude that if A is well formed, than B is well formed. The proof of the reverse direction is analogous and is omitted.

□

A.4 Proof of Theorem 5

Proof: Because Q is supported by the order preserving index, we know that $Od \prec_{\mathcal{T},D_Q} Od_Q$ and thus no sort is required. Also, because Tr is a balanced binary tree, we can traverse a path from the root node of the tree to any leaf node with a base two logarithmic number of nodes on the path. We now show that any conjunctive component of the query description will have a result which is a range of an in-order traversal of the description tree Tr. Disjunctive components are then a union of the ranges obtained from each of the u conjunctive components. This yields u logarithmic traversals of the tree $(u \cdot log(n))$, in addition to the size of the ranges corresponding to the query result (k) as the number of nodes, and thus subsumption tests, considered when evaluating a query.

Consider an arbitrary conjunctive component of the query description D, the set of descriptions subsumed by D are a range if the following two properties hold:

1. $\forall\langle D', Tr_1, Tr_2\rangle \in Tr$ such that $\mathcal{T} \models D' \sqsubseteq D$, $\neg(Od)_{\mathcal{T}}(D', D_1)$ and $\neg(Od)_{\mathcal{T}}(D_2, D')$.
2. $\forall\langle D', Tr_1, Tr_2\rangle \in Tr$ such that $\mathcal{T} \not\models D' \sqsubseteq D$, $\neg(Od)_{\mathcal{T}}(D_1, D')$ or $\neg(Od)_{\mathcal{T}}(D', D_2)$.

where D_1 is the left-most description in Tr, and D_2 is the right-most description in Tr, such that D_1 and D_2 are in the set of descriptions subsumed by D.

To prove the first property, assume that there exists a node $\langle D', Tr_1, Tr_2\rangle$ in Tr such that $\mathcal{T} \models D' \sqsubseteq D$, and either $(Od)_{\mathcal{T}}(D', D_1)$ or $(Od)_{\mathcal{T}}(D_2, D')$ for a proof by contradiction.

If $(Od)_{\mathcal{T}}(D', D_1)$, then we have a description D' which is left of D_1 in Tr (since Tr is well formed) and subsumed by D. This is a contradiction with D_1 being the left-most description in Tr subsumed by D. Similarly, if $(Od)_{\mathcal{T}}(D_2, D')$, then D' is right of D_2 in Tr and subsumed by D. However, this is a contradiction with D_2 being the right-most description in Tr subsumed by D. Thus the assumption that a node subsumed by D exists outside of the range fails.

To prove the second property we consider a structural induction on the ordering description Od.

- Base Case: $Od =$ "Un".
  Both $\neg(Od)_{\mathcal{T}}(D_1, D')$ and $\neg(Od)_{\mathcal{T}}(D', D_2)$ hold by the definition of an ordering.
- Inductive Hypothesis: Assume that $\forall\langle D', Tr_1, Tr_2\rangle \in Tr$ such that $\mathcal{T} \not\models D' \sqsubseteq D$, $\neg(Od_0)_{\mathcal{T}}(D_1, D')$ or $\neg(Od_0)_{\mathcal{T}}(D', D_2)$ for some arbitrary ordering description Od_0.
- Inductive Step: Consider an arbitrary ordering description Od. Assume that there exists a node $\langle D', Tr_1, Tr_2\rangle$ in Tr such that $\mathcal{T} \not\models D' \sqsubseteq D$, $(Od)_{\mathcal{T}}(D_1, D')$ and $(Od)_{\mathcal{T}}(D', D_2)$ for a proof by contradiction. Because D has no disjunctions, there are eight possibilities for D being sufficiently selective:
  1. Case: $\mathcal{T} \models \top \sqsubseteq D$.
     Trivially, $\mathcal{T} \models D' \sqsubseteq \top$, thus $\mathcal{T} \models D' \sqsubseteq D$ which is a contradiction.

2. Case: $Od =$ "$f : Od_1$", $SS_{\mathcal{T}}(E, Od_1)$ and $\mathcal{T} \models D \equiv ((f = k) \sqcap E)$. Because $\mathcal{T} \models D_1 \sqcap D_2 \sqsubseteq D$, we get $\mathcal{T} \models D_1 \sqcap D_2 \sqsubseteq ((f = k) \sqcap E)$ for some constant k and description E. Because D' is sufficiently descriptive, we know that $\mathcal{T} \models D' \sqsubseteq (f = k_2)$ for some constant k_2. Additionally, because $(Od)_{\mathcal{T}}(D_1, D')$, we can conclude that k_2 is greater than k, or they are equal and $(Od_1)_{\mathcal{T}}(D_1, D')$. Also, because $(Od)_{\mathcal{T}}(D', D_2)$, k_2 is less than k, or they are equal and $(Od_1)_{\mathcal{T}}(D', D_2)$. In the case where k and k_2 are equal, we know that $\neg(Od_1)_{\mathcal{T}}(D_1, D')$ and $\neg(Od_1)_{\mathcal{T}}(D', D_2)$ by the inductive hypothesis which yields a contradiction. In the case where k and k_2 are not equal, we have k being both less than and greater than k_2 which is also a contradiction.
3. Case: $Od =$ "$f : Od_1$" and $\mathcal{T} \models D \equiv (f < k)$.
   Because $\mathcal{T} \models D_2 \sqsubseteq D$ and D_2 is sufficiently descriptive, we know that $\mathcal{T} \models D_2 \sqsubseteq (f = k_1)$ for some constant k_1. Because $\mathcal{T} \models D \equiv (f < k)$, we can conclude that k_1 is less than k. Because D' is sufficiently descriptive, we know that $\mathcal{T} \models D' \sqsubseteq (f = k_2)$ for some constant k_2. Because $\mathcal{T} \not\models D' \sqsubseteq D$, k_2 is greater than or equal to k. Thus, k_1 is less than k_2 which is a contradiction with the assumption $(Od)_{\mathcal{T}}(D', D_2)$.
4. Case: $Od =$ "$f : Od_1$" and $\mathcal{T} \models D \equiv \neg(f < k)$.
   Because $\mathcal{T} \models D_1 \sqsubseteq D$ and D_1 is sufficiently descriptive, we know that $\mathcal{T} \models D_1 \sqsubseteq (f = k_1)$ for some constant k_1. Because $\mathcal{T} \models D \equiv \neg(f < k)$, we can conclude that k_1 is greater than or equal to k. Because D' is sufficiently descriptive, we know that $\mathcal{T} \models D' \sqsubseteq (f = k_2)$ for some constant k_2. Because $\mathcal{T} \not\models D' \sqsubseteq D$, k_2 is less than k. Thus, k_1 is greater than k_2 which is a contradiction with the assumption $(Od)_{\mathcal{T}}(D_1, D')$.
5. Case: $Od =$ "$f : Od_1$" and $\mathcal{T} \models D \equiv (\neg(f < k) \sqcap (f < k'))$.
   Because D' is sufficiently descriptive, we know that $\mathcal{T} \models D' \sqsubseteq (f = k_1)$ for some constant k_1. Also, because $\mathcal{T} \not\models D' \sqsubseteq D$, k_1 is either less than k, or greater than or equal to k'. In either case the reasoning from the previous two cases apply in the exact same manner.
6. Case: $Od =$ "$D_0(Od_1, Od_2)$", $SS_{\mathcal{T}}(E, Od_1)$ and $\mathcal{T} \models D \equiv (D_0 \sqcap E)$.
   Because $\mathcal{T} \models D_1 \sqcap D_2 \sqsubseteq D$, and $\mathcal{T} \models D \sqsubseteq D_0$, we know that $\mathcal{T} \models$

$D_1 \sqcap D_2 \sqsubseteq D_0$. Because $(Od)_{\mathcal{T}}(D', D_2)$, it must be the case that $\mathcal{T} \models D' \sqsubseteq D_0$. Thus $(Od_1)_{\mathcal{T}}(D_1, D')$ and $(Od_1)_{\mathcal{T}}(D', D_2)$ because of the initial assumption of $(Od)_{\mathcal{T}}(D_1, D')$ and $(Od)_{\mathcal{T}}(D', D_2)$. However, $\neg(Od_1)_{\mathcal{T}}(D_1, D')$ and $\neg(Od_1)_{\mathcal{T}}(D', D_2)$ by the inductive hypothesis which is a contradiction.

7. Case: $Od =$ "$D_0(Od_1, Od_2)$", $SS_{\mathcal{T}}(E, Od_2)$ and $\mathcal{T} \models D \equiv (\neg D_0 \sqcap E)$.
   Because $\mathcal{T} \models D_1 \sqcap D_2 \sqsubseteq D$, and $\mathcal{T} \models D \sqsubseteq \neg D_0$, we know that $\mathcal{T} \models D_1 \sqcap D_2 \sqsubseteq \neg D_0$. Because $(Od)_{\mathcal{T}}(D_1, D')$, it must be the case that $\mathcal{T} \models D' \sqsubseteq \neg D_0$. Thus $(Od_2)_{\mathcal{T}}(D_1, D')$ and $(Od_2)_{\mathcal{T}}(D', D_2)$ because of the initial assumption of $(Od)_{\mathcal{T}}(D_1, D')$ and $(Od)_{\mathcal{T}}(D', D_2)$. However, $\neg(Od_2)_{\mathcal{T}}(D_1, D')$ and $\neg(Od_2)_{\mathcal{T}}(D', D_2)$ by the inductive hypothesis which is a contradiction.

By induction, the assumption of a description which is not subsumed by D existing in the range fails.

Thus, the total number of comparisons (and thus subsumption tests) is $O(k + u \cdot lg(n))$. □

Appendix B

Experimental Data

B.1 Workload

The following XQueries labeled Q_1 to Q_{14} are from the DC/SD XBench benchmark [2]. Q_{21} and Q_{22} are contributed by the author.

Q1
```
for $item in input()/catalog/:item[@id="I1"]
return $item
```

Q2
```
for $item in input()/catalog/:item
where $item/authors/author/name/first_name = "Ben"
return $item/title
```

Q5
```
for $a in input()/catalog/:item[@id="I3"]
return $a/authors/author[1]
```

Q6
```
for $item in input()/catalog/:item
where some $auth in
$item/authors/author/contact_information/mailing_address
satisfies $auth/name_of_country = "Canada"
return $item
```

Q8
```
for $a in input()/catalog/*[@id="I4"]
return $a/publisher
```

Q9
```
for $a in input()/catalog/:item
where $a/@id="I5"
return $a//ISBN/text()
```

Q12
```
for $a in input()/catalog/:item[@id="I6"]
return
⟨Output⟩
$a/authors/author[1]/contact_information/mailing_address
⟨/Output⟩
```

Q14
```
for $a in input()/catalog/:item
where $a/date_of_release gt "1990-01-01" and
$a/date_of_release lt "1991-01-01" and
empty($a/publisher/contact_information/FAX_number)
return
⟨Output⟩ $a/publisher/name
⟨/Output⟩
```

Q21
```
⟨RESULTS⟩ {
for $item in /catalog/item
where $item/pricing/when_is_available lt "1990-01-01"
order by $item/pricing/suggested_retail_price
return $item,
for $item in /catalog/item
where $item/pricing/when_is_available  gt "1990-01-01"
order by $item/pricing/cost
return $item }
⟨RESULTS⟩
```

Q22

```
for $item in /catalog/item
where ($item/author/mailing_address/name_of_state ="New York"
or $item/publisher/mailing_address/name_of_state="New York" )
and $item/release_date gt "1995-01-01"
and $item/release_date lt "2005-01-01"
return $item
```

B.2 Schema

The XBench [2] DC/SD schema.

```
⟨?xml version="1.0" encoding="UTF-8"?⟩
⟨!ELEMENT FAX_number (#PCDATA)⟩
⟨!ELEMENT ISBN (#PCDATA)⟩
⟨!ELEMENT attributes (ISBN, number_of_pages, type_of_book, size_of_book)⟩
⟨!ELEMENT author (name, date_of_birth, biography, contact_information)⟩
⟨!ELEMENT authors (author+)⟩
⟨!ELEMENT biography (#PCDATA)⟩
⟨!ELEMENT catalog (item+)⟩
⟨!ATTLIST catalog
xmlns:xsi CDATA #REQUIRED
xsi:noNamespaceSchemaLocation CDATA #REQUIRED ⟩
⟨!ELEMENT contact_information (mailing_address, FAX_number?,
phone_number, email_address?, web_site?)⟩
⟨!ELEMENT cost (#PCDATA)⟩
⟨!ATTLIST cost
currency CDATA #REQUIRED ⟩
⟨!ELEMENT country (name, exchange_rate, currency)⟩
⟨!ELEMENT currency (#PCDATA)⟩
⟨!ELEMENT data (#PCDATA)⟩
⟨!ELEMENT date_of_birth (#PCDATA)⟩
⟨!ELEMENT date_of_release (#PCDATA)⟩
⟨!ELEMENT description (#PCDATA)⟩
⟨!ELEMENT email_address (#PCDATA)⟩
⟨!ELEMENT web_site (#PCDATA)⟩
⟨!ELEMENT exchange_rate (#PCDATA)⟩
⟨!ELEMENT first_name (#PCDATA)⟩
⟨!ELEMENT height (#PCDATA)⟩
⟨!ATTLIST height
unit CDATA #REQUIRED ⟩
⟨!ELEMENT image (data)⟩
⟨!ELEMENT item (title, authors, date_of_release, publisher,
subject, description, related_items, media, pricing, attributes)⟩
⟨!ATTLIST item
id ID #REQUIRED ⟩
```

```
⟨!ELEMENT item_id (#PCDATA)⟩
⟨!ELEMENT last_name (#PCDATA)⟩
⟨!ELEMENT length (#PCDATA)⟩
⟨!ATTLIST length
unit CDATA #REQUIRED ⟩
⟨!ELEMENT mailing_address (street_information, name_of_city,
name_of_state, zip_code, name_of_country?, country?)⟩
⟨!ELEMENT media (thumbnail, image)⟩
⟨!ELEMENT middle_name (#PCDATA)⟩
⟨!ELEMENT name (#PCDATA | first_name | middle_name | last_name)*⟩
⟨!ELEMENT name_of_city (#PCDATA)⟩
⟨!ELEMENT name_of_country (#PCDATA)⟩
⟨!ELEMENT name_of_state (#PCDATA)⟩
⟨!ELEMENT number_of_pages (#PCDATA)⟩
⟨!ELEMENT phone_number (#PCDATA)⟩
⟨!ELEMENT pricing (suggested_retail_price, cost, when_is_available,
quantity_in_stock)⟩
⟨!ELEMENT publisher (name, contact_information)⟩
⟨!ELEMENT quantity_in_stock (#PCDATA)⟩
⟨!ELEMENT related_item (item_id)⟩
⟨!ELEMENT related_items (related_item+)⟩
⟨!ELEMENT size_of_book (length, width, height)⟩
⟨!ELEMENT street_address (#PCDATA)⟩
⟨!ELEMENT street_information (street_address+)⟩
⟨!ELEMENT subject (#PCDATA)⟩
⟨!ELEMENT suggested_retail_price (#PCDATA)⟩
⟨!ATTLIST suggested_retail_price
currency CDATA #REQUIRED ⟩
⟨!ELEMENT thumbnail (data)⟩
⟨!ELEMENT title (#PCDATA)⟩
⟨!ELEMENT type_of_book (#PCDATA)⟩
⟨!ELEMENT when_is_available (#PCDATA)⟩
⟨!ELEMENT width (#PCDATA)⟩
⟨!ATTLIST width
unit CDATA #REQUIRED ⟩
⟨!ELEMENT zip_code (#PCDATA)⟩
```

Bibliography

[1] Walid Aref and Ihab Ilyas. SP-GiST: An Extensible Database Index for Supporting Space Partitioning Trees. *Journal of Intelligent Information Systems*, 17(2-3):215–240, December 2001.

[2] B. B. Yao, M. T. Ozsu, and N. Khandelwal. XBench Benchmark and Performance Testing of XML DBMSs. In *IEEE International Conference on Data Engineering*, pages 621–632, 2004.

[3] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. *The Description Logic Handbook: Theory, Implementation, and Applications.* Cambridge University Press, 2003.

[4] Holger Bast, Alexandru Chitea, Fabian Suchanek, and Ingmar Weber. ESTER: efficient search on text, entities, and relations. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 671–678, New York, NY, USA, 2007. ACM.

[5] S. Bechhofer, R. Moller, and P. Crowther. The DIG Description Logic Interface. In *In Proc. of International Workshop on Description Logics (DL2003)*, 2003.

[6] Alexander Borgida, Ronald J. Brachman, Deborah L. McGuinness, and Lori Alperin Resnick. CLASSIC: A Structural Data Model for Objects. In *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, pages 58–67. ACM Press, 1989.

[7] Alexander Borgida, Ronald J. Brachman, Deborah L. McGuinness, and Lori Alperin Resnick. Classic: A structural data model for objects. In James Clifford, Bruce G. Lindsay, and David Maier, editors, *Proceedings*

of the 1989 ACM SIGMOD International Conference on Management of Data, Portland, Oregon, May 31 - June 2, 1989, pages 58–67. ACM Press, 1989.

[8] Ronald J. Brachman and James G. Schmolze. An overview of the kl-one knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[9] Jeen Broekstra, Arjohn Kampman, and Frank van Harmelen. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In *ISWC '02: Proceedings of the First International Semantic Web Conference on The Semantic Web*, pages 54–68, London, UK, 2002. Springer-Verlag.

[10] Jeremy J. Carroll, Ian Dickinson, Chris Dollin, Dave Reynolds, Andy Seaborne, and Kevin Wilkinson. Jena: implementing the semantic web recommendations. In *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 74–83, New York, NY, USA, 2004. ACM.

[11] DBpedia. `http://dbpedia.org`.

[12] Dmitry Tsarkov and Ian Horrocks. FaCT++ Description Logic Reasoner: System Description . In *3rd International Joint Conference on Automated Reasoning*, 2006.

[13] Freebase. `http://www.freebase.com`.

[14] Galax. `http://www.galaxquery.org/`.

[15] Nicholas Gibbins, Stephen Harris, and Nigel Shadbolt. Agent-based semantic web services. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 1(2):141–154, 2003.

[16] R. Guha, Rob McCool, and Eric Miller. Semantic search. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 700–709, New York, NY, USA, 2003. ACM.

[17] Antonin Guttman. R-trees: a dynamic index structure for spatial searching. In *SIGMOD '84: Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, pages 47–57, New York, NY, USA, 1984. ACM.

[18] Stephen Harris and Nicholas Gibbins. 3store: Efficient Bulk RDF Storage. In *In Proceedings of the 1st International Workshop on Practical and Scalable Semantic Systems*, volume 89 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2003.

[19] Joseph Hellerstein, Jeffrey Naughton, and Avi Pfeffer. Generalized Search Trees for Database Systems. In *International Conference on Very Large Data Bases*, pages 562–573, 1995.

[20] Atanas Kiryakov, Damyan Ognyanov, and Dimitar Manov. OWLIM - A Pragmatic Semantic Repository for OWL. In *WISE Workshops*, volume 3807 of *Lecture Notes in Computer Science*, pages 182–192. Springer, 2005.

[21] Jeffrey Pound, Lubomir Stanchev, David Toman, and Grant Weddell. On Ordering Descriptions in a Description Logic. *20th International Workshop on Description Logics*, pages 123–134, 2007.

[22] Jeffrey Pound, Lubomir Stanchev, David Toman, and Grant Weddell. On Ordering Descriptions in a Description Logic. Technical Report CS-2007-16, David R. Cheriton School of Computer Science, University of Waterloo, 2007.

[23] Qexo. `http://www.gnu.org/software/qexo/`.

[24] RDF Vocabulary Description Language 1.0: RDF Schema. `http://www.w3.org/TR/rdf-schema/`.

[25] Resource Description Framework (RDF). `http://www.w3.org/RDF/`.

[26] Hanan Samet. The Quadtree and Related Hierarchical Data Structures. *ACM Comput. Surv.*, 16(2):187–260, 1984.

[27] David E. Simmen, Eugene J. Shekita, and Timothy Malkemus. Fundamental Techniques for Order Optimization. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 57–67, 1996.

[28] SPARQL Query Language for RDF. `http://www.w3.org/TR/rdf-sparql-query/`.

[29] Lubomir Stanchev and Grant Weddell. Index Selection for Embedded Control Applications using Description Logics. In *Description Logics 2003*, pages 9–18. CEUR-WS vol.81, 2003.

[30] Stelios Paparizos and H.V. Jagadish. Pattern Tree Algebras: Sets or Sequences? In *International Conference on Very Large Data Bases*, pages 349–360, 2005.

[31] Heiner Stuckenschmidt and Jeen Broekstra. Time - Space Trade-Offs in Scaling up RDF Schema Reasoning. In *Web Information Systems Engineering - WISE Workshops*, pages 172–181, 2005.

[32] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A Core of Semantic Knowledge - Unifying WordNet and Wikipedia. In Carey L. Williamson, Mary Ellen Zurko, and Prashant J. Patel-Schneider, Peter F. Shenoy, editors, *16th International World Wide Web Conference (WWW 2007)*, pages 697–706, Banff, Canada, 2007. ACM.

[33] SWRL: A Semantic Web Rule Language. `http://www.w3.org/Submission/SWRL/`.

[34] The Gene Ontology. `http://www.geneontology.org`.

[35] Raphael Volz, Steffen Staab, and Boris Motik. Incrementally Maintaining Materializations of Ontologies Stored in Logic Databases. *Journal of Data Semantics II*, 3360:1–34, 2005. LNCS, Springer.

[36] Web Ontology Language (OWL). `http://www.w3.org/2004/OWL/`.

[37] Gang Wu and Juanzi Li. Managing Large Scale Native RDF Semantic Repository from the Graph Model Perspective. In *Proceedings of SIGMOD 2007 Ph.D. Workshop on Innovative Database Research (IDAR2007)*, 2007.

[38] X-Hive/DB. `http://www.x-hive.com`.

[39] Xerces XML Parser. `http://xerces.apache.org/xerces-j/`.

[40] Qingwei Xu, Yixiang Shi, Qiang Lu, Guoqing Zhang, Qingming Luo, and Yixue Li. GORouter: an RDF model for providing semantic query and inference services for Gene Ontology and its associations. *BMC Bioinformatics*, 9, 2008.