Approximating Terminological Queries

Heiner Stuckenschmidt, Frank van Harmelen

Vrije Universiteit Amsterdam
de Boelelaan 1081a,
1081 HV Amsterdam, The Netherlands
email: {heiner,frankh}@cs.vu.nl

Abstract. Current proposals for languages to encode terminological knowledge in intelligent systems support logical reasoning for answering user queries about objects and classes. An application of these languages on the World Wide Web, however, is hampered by the limitations of logical reasoning in terms of efficiency and flexibility. In this paper we describe, how techniques from approximate reasoning can be used to overcome these problems. We discuss terminological knowledge and approximate reasoning in general and show the benefits of approximate reasoning using the example of building and maintaining semantic catalogues that can be used to query resource locations based on object classes.

1 Motivation

Recent research on the so-called Semantic Web aims at providing models and methods to make information on the World Wide Web accessible to machines instead of humans users. This aim requires a precise semantic description of information items, because different from human users, machines are not able to distinguish relevant from irrelevant in the absence of a precise description of its meaning. It has been argued that ontologies are a technology that provides such precise definitions of the terminology used in information sources [5]. Consequently much work has been done on the development of an infrastructure for representing and reasoning about ontologies on the web. Languages such as OIL (building on RDF Schema [1], later extended into DAML+OIL [13] and currently being standardized by W3C as OWL) used to encode terminological knowledge and advanced reasoning systems like RACER [7] exist that can be used to check and query terminological knowledge.

A strength of the current proposals for the foundational languages of the Semantic Web (RDF Schema, DAML+OIL), it that they are all based on formal logic. However, this reliance on logics is not only a strength but also a weakness. Traditionally, logic has always aimed at modeling idealized forms of reasoning under idealized circumstances. Clearly, this is not what is required under the practical circumstances of the Semantic Web. In a distributed and heterogeneous environment like the World Wide Web, we face a situation, where ontologies are built by many people that are not experts in knowledge modeling independent of each other. The resulting models will often lack precision or even contain contradictory information. These problems together with the intimidating size of the Web leads to the need for reasoning under time-pressure, reasoning with other limited resources besides time (e.g. limited memory, incomplete knowledge), reasoning that is not "perfect" but instead "good enough" for given tasks under given circumstances and algorithms that do not behave as yes/no oracles, but that instead display anytime behaviour [3]. It is tempting to conclude that symbolic, formal logic fails on all these counts, and to abandon that paradigm. However, research in the past few years has developed approximate reasoning methods with the above properties while staying within the framework of symbolic, formal logic [10, 14, 2, 12].

In this paper we propose an approach for approximating the answers to terminological queries based on non-equivalent transformations of increasing exactness. We start with a general definition of terminological knowledge. We then discuss queries over terminological knowledge and review the approach of [9] for answering such queries. As the main contribution of the paper we describe different strategies for approximating answers to terminological queries by modifying the query answering approach mentioned above. We summarize with a discussion of open problems and further options for computing approximate answers.

2 Preliminarities: Terminological Knowledge

Before we can discuss the application of approximate reasoning techniques to terminological knowledge on the World Wide Web, we first have to clarify our view on terminological knowledge. We do this by defining terminological knowledge bases on a level concrete enough to precisely define reasoning tasks and abstract enough to be independent of a concrete language. These definitions are consistent with models of terminological knowledge based on Description Logics [4], which are underlying much of the current work on ontology languages for the Semantic Web such as OIL, DAML+OIL and OWL Based on this definition, we define a number of terminological queries, we might want to state and give a general notion of approximate results of such queries.

2.1 Knowledge Bases

A number of languages for encoding terminological knowledge on the Web have been proposed (see [6] for an overview). In order to get a general notion of terminological knowledge, we define the general structure of a terminological knowledge base independent of a concrete language.

Definition 1 (Terminological Knowledge Base). *A Terminological Knowledge Base $\mathcal{T}$ is a triple*

$$\mathcal{T} = \langle \mathcal{C}, \mathcal{R}, \mathcal{O} \rangle$$

where $\mathcal{C}$ is a set of class definitions, $\mathcal{R}$ is a set of relation definitions and $\mathcal{O}$ is a set of object definitions.

Terminological knowledge usually groups objects of the world that have certain properties in common (e.g. cities or countries). A description of the shared properties is called a class definition. Concepts can be arranged into a subclass-superclass relation in order to be able to further discriminate objects into subgroups (e.g. capitals or European countries). Classes can be defined in two ways, by enumeration of its members or by stating that it is a refinement of a complex logical expressions. The specific logical operators to express such logical definitions can vary between ontology languages; the general definitions we give here abstract from these specific operators.

Definition 2 (Class Definitions). *A class definition is an axiom of one of the following forms:*

- $c \equiv (o_1, \cdots, o_n)$ *where c is a class definition and $o_1, \cdots, o_n$ are object definitions.*
- $c_1 \sqsubseteq c_2$ *where c_1 and c_2 are class definitions.*

Further, there is the universal class denoted as $\top$.

Objects of the same type normally occur in similar situations where they have a certain relation to each other (cities lie in countries, countries have a capital). These typical relations can often be specified in order to establish structures between classes. Terminological knowledge considers binary relations that can either be defined by restricting their domain and range or by declaring it to be a sub-relation of an existing one.

Definition 3 (Relation Definitions). *A relation definition is an axiom of one of the following forms:*

- $r \sqsubseteq (c_1, c_2)$ *where r is a role definition and c_1 and c_2 are class definitions.*
- $r_1 \sqsubseteq r_2$ *where r_1 and r_2 are role definitions.*

The universal role is defined as $\top \times \top$.

Sometimes single objects (e.g. the continent Europe) play a prominent role in a domain of interest, or the membership of a concept is defined by the relation to a specific object (European countries are those contained in Europe). For this purpose ontology languages often allow to specify single objects, also called instances. In our view on terminological knowledge, instances can be defined by stating their membership in a class. Further, we can define instances of binary relations by stating that two objects form such a pair.

Definition 4 (Object Definitions). *An object definition is an axiom of one of the following forms:*

- $o : c$ *where* c *is a class definition and* o *is an individual.*
- $(o_1, o_2) : r$ *where* r *is a relation definition and* o_1, o_2 *are object definitions.*

In the following, we will consider terminological knowledge bases that consist of such axioms. Of course, any specific ontology language will have to further instantiate these definitions to specify logical operators between classes etc, but for the purposes of this paper, these general definitions are sufficient. Further, we define the signature of a terminological knowledge base to be a triple $\langle \mathcal{CN}, \mathcal{RN}, \mathcal{IN} \rangle$, where $\mathcal{CN}$ is the set of all names of classes defined in $\mathcal{C}$, $\mathcal{RN}$ the set of all relation names and $\mathcal{IN}$ the set of all object names occurring in the knowledge base.

2.2 Semantics and Logical Consequence

We can define semantics and logical consequence of a terminological knowledge base using an interpretation mapping $.^{\Im}$ into an abstract domain Δ such that:

- $c^{\Im} \subseteq \Delta$ for all class definitions c in the way defined above
- $r^{\Im} \subseteq \Delta \times \Delta$ for all relation definition r
- $o^{\Im} \in \Delta$ for all object definitions o

This type of denotational semantics is inspired by description logics [4], however, we are not specific about operators that can be used to build class definitions which are of central interest of these logics. Using the interpretation mapping, we can define the notion of a model in the following way:

Definition 5 (Model of a Terminological Knowledge Base). *An interpretation* $\Im$ *is a model for the knowledge base* $\mathcal{T}$ *if* $\Im \models A$ *for every axiom* $A \in (\mathcal{C} \cup \mathcal{R} \cup \mathcal{O})$ *where* $\models$ *is defined as follows.*

- $\Im \models c \equiv (o_1, \cdots, o_n)$, *iff* $c^{\Im} = \{o_1^{\Im}, \cdots, o_n^{\Im}\}$
- $\Im \models c_1 \sqsubseteq c_2$, *iff* $c_1^{\Im} \subseteq c_2^{\Im}$
- $\Im \models r \sqsubseteq (c_1, c_2)$, *iff* $r^{\Im} \subseteq c_1^{\Im} \times c_2^{\Im}$
- $\Im \models r_1 \sqsubseteq r_2$, *iff* $r_1^{\Im} \subseteq r_2^{\Im}$
- $\Im \models o : c$, *iff* $o^{\Im} \in c^{\Im}$
- $\Im \models (o_1, o_2) : r$, *iff* $(o_1^{\Im}, o_2^{\Im}) \in r^{\Im}$

Having defined the notion of model, logical consequence can be defined in a straightforward way:

Definition 6 (Logical Consequence). *An axiom* A *logically follows from a set of axioms* $\mathcal{S}$ *if* $\Im \models \mathcal{S}$ *implies* $\Im \models A$ *for every model* $\Im$*. We denote this fact by* $\mathcal{S} \models A$*.*

Our work heavily relies on this notion of model and logical consequence, because query results are determined with respect to being the logical consequence of a certain set of axioms. Further, we need the notion of a model in order to characterize approximate results.

3 Querying Terminological Knowledge

In the following we first define ontology based queries as well as the notion of answers to and relations between queries. We formalize queries in the following way: conjuncts of a query are predicates that correspond to classes and relations of the ontology. Further, variables in a query may only be instantiated by constants that correspond to objects in that ontology.

Definition 7 (Terminological Queries). *Let V be a set of variables disjoint from $\mathcal{IN}$ then an terminological query Q over a knowledge base $\mathcal{T}$ is an expressions of the form*

$$Q \leftarrow q_{1_i} \wedge \cdots \wedge q_{m_i}$$

where q_i are query terms of the form $x : C$ or $(x, y) : R$ such that $x, y \in V \cup \mathcal{IN}$, $C \in \mathcal{CN}$ and $R \in \mathcal{RN}$.

We use the following toy example of a terminological knowledge base to illustrate their approach and the extensions we propose:

$$\begin{aligned}
Human &\sqsubseteq \top & (1)\\
Male &\sqsubseteq Human & (2)\\
Female &\sqsubseteq Human & (3)\\
Organization &\sqsubseteq \top & (4)\\
University &\sqsubseteq Organization & (5)\\
Title &\sqsubseteq \top & (6)\\
father\text{-}of &\sqsubseteq Male \times Human & (7)\\
works\text{-}at &\sqsubseteq Human \times Organization & (8)\\
degree &\sqsubseteq Human \times Title & (9)\\
granted\text{-}by &\sqsubseteq Title \times University & (10)\\
vu &: University & (11)
\end{aligned}$$

In particular, we consider the following query formulated over the example knowledge base:

$$\begin{aligned}
Q(X) \leftarrow\ &(X, Y) : father\text{-}of \wedge (Y, Z) : works\text{-}at \wedge Y : Female \wedge \\
&(Y, V) : degree \wedge (V, vu) : awarded\text{-}by \qquad (12)
\end{aligned}$$

This query asks for all persons that are father of a female person that has a degree awarded by the Vrije Universiteit Amsterdam and works somewhere.

The fact that all conjuncts relate to elements of the ontology allows us to determine the answer to terminological queries in terms of instantiations of the query that are logical consequences of the knowledge base it refers to:

Definition 8 (Query Answers). *The answer of a query Q containing variables $v_1, \cdots, v_k$ over a knowledge base T is a set of tuples $(i_1, \cdots, i_k)$ such that $T \models Q'$ where Q' is the query obtained from Q by substituting $v_1, \cdots, v_k$ by $i_1, \cdots, i_k$. The projections of the answer tuples to variables occurring in the head of a query Q is denoted as $res(Q)$ and referred to as the answer set of Q.*

Based on the possible answers to a query, we can define semantic relations between different queries we will later use to characterize approximations. In particular, we consider query containment and query equivalence (compare [8])

Definition 9 (Query Containment and Equivalence). *Let $\mathcal{T} = \langle \mathcal{C}, \mathcal{R}, \mathcal{O} \rangle$ and Q_1, Q_2 conjunctive queries over $\mathcal{T}$. Q_1 is said to be contained in another query Q_2 denoted by $Q_1 \sqsubseteq Q_2$ if for all possible sets of object definitions of a terminological knowledge base the answers for Q_1 is a subset of the answers for Q_2 : $(\forall \mathcal{O} : res(Q_1) \subseteq res(Q_2))$. The two queries are said to be equivalent, denoted as $Q_1 \equiv Q_2$ iff $Q_1 \sqsubseteq Q_2$ and $Q_2 \sqsubseteq Q_1$*

An example for query subsumption is the following query that can be shown to subsume the example query above:

$$\begin{aligned} Q'(X) \leftarrow X : Male \wedge (X, Y) : father\text{-}of \wedge Y : Female \wedge \\ (Y, Z) : degree \wedge (Z, U) : awarded\text{-}by \wedge U : University \end{aligned} \tag{13}$$

There are three points where this query differs from Q. First of all the awarding organization is not restricted to a specific one, but only to universities in general. Further, the conjunct concerning the *works-at* relation is missing. For these two changes, it is easy to see that the set on possible answers includes all the answers of Q. The last difference, which is the additional information that X should be of type male seems to be more restrictive than the corresponding restriction in Q. However, as the $father\text{-}of$ relation is defined over the domain of male persons, this additional conjunct does not change the set of possible answers.

Based on these semantic relation between queries, we can distinguish three different kinds of transformations, namely equivalent transformations resulting in a query that is equivalent to the original query, contained and containing transformations.

Definition 10 (Query Transformations). *Let Q and Q' be queries over the same Knowledge base, then*

- *A query Q' is an equivalent transformation of Q if $Q \equiv Q'$.*
- *A query Q' is a contained transformation of Q if $Q \sqsubseteq Q'$*
- *A query Q' is a containing transformation of Q if $Q' \sqsubseteq Q$*

In the following section, we discuss a method for actually computing the result of a conjunctive query over a terminology. We restrict ourselves to queries with only one variable in the query head.

4 Approximating Conjunctive Queries

The underlying assumption of our approach is that less complex queries can be answered in shorter time. Following this idea, we propose to compute a sequence $Q^1, \cdots, Q^n$ of queries starting with very simple ones while gradually increasing their complexity. In order to use these simpler queries as approximations for the original query we have to ensure the following properties of the sequence of queries:

1. $i < j \implies Q^i \sqsupseteq Q^j$
2. $Q_n \equiv Q$

The first property ensures that the quality of the results of the queries is not decreasing. The second claims that the last query computed returns the desired exact result. Together, these properties ensure that the sequence of queries approximate the exact result.

Next, we have to define how to determine queries to be used for the approximation. This process is constrained by the two properties above. We decide to start with the universal query Q^0 that returns all objects in the knowledge base and to successively add conjuncts from the original query. It is easy to see that this gives us the desired properties as adding a conjunct leads to a subsumed query. Further, as the original query has a finite number of conjuncts, adding conjuncts necessarily leads to the original query after a finite number of steps. The critical part concerning this approach is the step of deciding which conjunct(s) to add next in order to generate the next query in the sequence. This choice has a strong influence on the quality of the approximation as a badly chosen ordering may lead to a situation, were the answer set remains the same for a long time and only reduces to the exact answer in the last step. In the following, we discuss different strategies for determining sequences of subsuming queries that try to avoid the problem mentioned.

4.1 Node Expansion Approximation

An approximation that actually increases the quality of the result in every step is considered a good approximation. In order to achieve such an approximation, we have to try to specify the next query in the sequence in such a way that it excludes objects from the current answer set. In order to produce good approximations and not just arbitrary ones, we have to make sure that the next conjuncts added to the query expression directly apply to the objects in the answer set. This in turn depends on the dependencies between the variables in the query. In order to keep track of such dependencies Horrocks and Tessaris [9] introduce the notion of a query graph.

Definition 11 (Query Graph (Horrocks and Tessaris 2000)). *The graph induced by a query is a directed graph with a node for every variable and individual name in the query and an directed edge from node x to node y for every role term $(x, y) : R$ in the query.*

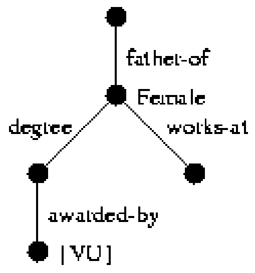

Fig. 1. Query Graph of the Example Query

While the approach of Horrocks and Tessaris is more general, we restrict ourselves to queries where the query graph is a (directed) tree and its root node corresponds to the variable we are interested in. In particular, this requires that none of the roles used in the query is declared to be functional and that each constant only appears once in a query. We can use the query graph as a basic structure to determine the next query in the sequence. In especially, we can start with the query that corresponds to the node of the query graph representing the answer variable. Starting from this node, we then successively expand nodes by adding their children and the corresponding arcs. Figure 2 shows the query graphs corresponding to the sequence of queries we obtain for the example query by applying the node expansion strategy.

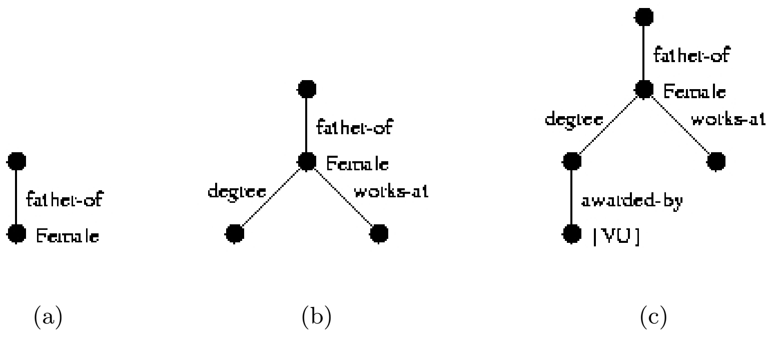

Fig. 2. Sequence of Query-Graphs for Node Expansion Approximation

The intuition of this approach is that in every step, we select one variable from the query and apply add all restrictions from the original query that apply to this specific variable. Applying these restrictions makes it very likely that the set of objects that can be used to instantiate this variable is reduced. The structure of the query graph ensures that this restriction has a potential influence on the set of objects we are interested in. Therefore it is also likely that the set of possible answers to the query is reduced.

We can also relate this back to the original idea of successively adding conjuncts to the universal query until we get the original query. The sequence of queries described above relates to the following series of conjunctive queries. We replace re-occurring conjuncts form previous queries in the sequence by the name of the corresponding query for better readability.

- $Q^1(X) \leftarrow (X,Y) : \textit{father-of} \wedge Y : \textit{Female}$
- $Q^2(X) \leftarrow Q^1(X) \wedge (Y,Z) : \textit{works-at} \wedge (Y,V) : \textit{degree}$
- $Q^3(X) \leftarrow Q^2(X)(V,vu) : \textit{awarded-by}$

We can easily see that this sequence of queries satisfies the requirements as they are strict extensions of each other and Q^3 is equal to Q.

4.2 Arc-Based Approximations

A potential problem of the node expansion strategy presented above is the granularity of the modifications made in each step of the approximation process. The approach of adding all restrictions on a variable at once may be problematic, because the change in the answer set may be radical, especially if many restrictions to a single variable are contained in the query. Adding many restriction at once will also often lead to a very small number of possible approximations. in fact, in this case the number of approximations is equal to the number of distinct variables in the query. In general, however, the number of possible approximations is equal to the number of possible subtrees of the original query trees. In order to overcome this problem, we now consider approximation strategies that ensure that only a limited number of restrictions are added at a time. This is achieved by only expanding the query graph one arc at a time. Using this approach, choosing the next expansion step is even more difficult, because we do not only have to decide which node to expand, but also which arc to add to that node. In the following we introduce two general strategies for selecting arcs to expand based on well-known search techniques.

Breadth-First Approximation: The first possibility to apply arc-based approximations is to adopt a breath-first strategy where first all arcs on the highest level of the tree are expanded one at a time before moving to the next level. The corresponding sequence of query graphs is shown in figure 3

The intuition of this strategy is to first restrict variables that are closely related to the goal variable. The assumption is that the instantiation of these variables have a stronger influence on the objects in the query set.

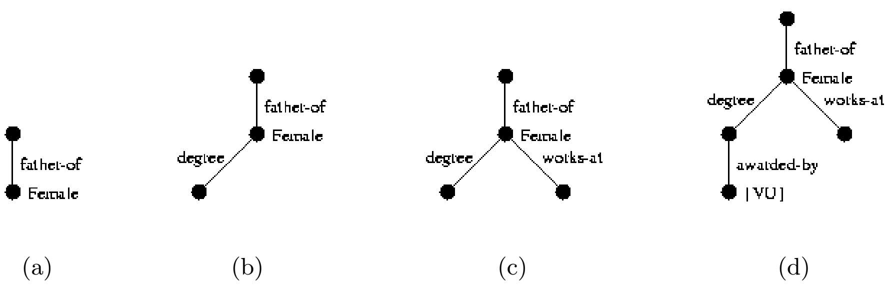

Fig. 3. Sequence of Query-Graphs for Breath-First Approximation

Looking at the corresponding conjunctive queries, we see that only one conjunct containing a binary predicate and therefore restricting the property of an object is added at each step of the approximation.

- $Q^1(X) \leftarrow (X,Y) : father\text{-}of \wedge Y : Female$
- $Q^2(X) \leftarrow Q^1(X) \wedge (Y,V) : degree$
- $Q^3(X) \leftarrow Q^2(X) \wedge (Y,Z) : works\text{-}at$
- $Q^4(X) \leftarrow Q^3(X) \wedge (V,vu) : awarded\text{-}by$

Depth-First Approximation: Analogously, we can apply a depths-first strategy for determining the next expansion of the query tree, In this case nodes at the deepest level of the partially expanded tree are expanded first. After the deepest level is reached, the expansion is continued at the second lowest level of the same branch and so on. The corresponding sequence of query trees is shown in figure 4.

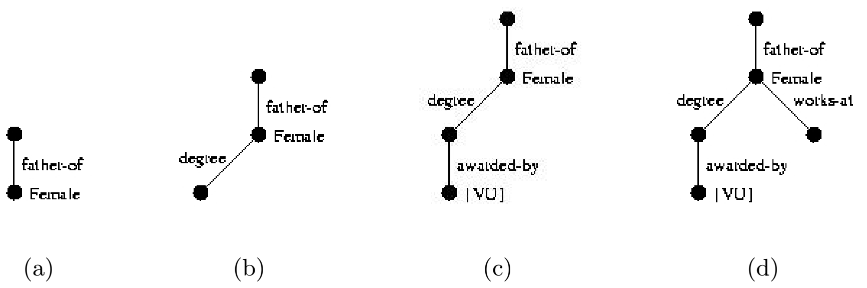

Fig. 4. Sequence of Query-Graphs for Depth-First Approximation

The intuition behind this rather strange looking strategy is to give a preference to role chains occurring the data. This choice is motivated by the assumption that long role chains are rather an exception and therefore provide a strong criterion for excluding objects from the answer set early in the approximation process.

In the corresponding sequence of conjunctive queries we can observe a preference for first introducing new variables before further restricting the old ones. As a consequence, a larger number of variables is restricted early in the approximation process. This also makes a significant reduction of possible answers in the early stages of the approximation likely.

- $Q^1(X) \leftarrow (X, Y) : father\text{-}of \wedge Y : Female$
- $Q^2(X) \leftarrow Q^1(X) \wedge (Y, V) : degree$
- $Q^3(X) \leftarrow Q^2(X) \wedge (V, vu) : awarded\text{-}by$
- $Q^4(X) \leftarrow Q^3(X) \wedge (Y, Z) : works\text{-}at$

5 Query Execution

Actually executing conjunctive queries over terminologies is a difficult task because unlike conjunctive queries over a database, there is not a single minimal model that satisfies the query. Due to this fact, query answering in the face of complex terminologies needs deductive reasoning. In the following, we describe the approach for answering conjunctive queries over terminologies proposed by Horrocks and Tessaris [9]. The idea of the approach of Horrocks and Tessaris is to translate the query into an equivalent concept expression, classify this new concept and use standard inference methods to check whether an object is an instance of the concept corresponding to the query expression. This approach makes use of the fact that binary relations in a conjunctive query can be translated into an existential restriction in such a way that logical consequence is preserved after a minor modification of the A-Box. Details are given in the following theorem.

Theorem 1 (Role Roll-Up (Horrocks and Tessaris 2000)). *Let $\langle \mathcal{C}, \mathcal{R}, \mathcal{O} \rangle$ be a terminological knowledge base with T-Box T and A-Box A. Let furtherR be a role, C_I Concept names in T and a, b be individual names in A. Given a new concept name P_b not appearing in T, then*

$$\langle \mathcal{C}, \mathcal{R}, \mathcal{O} \rangle \models (a, b) : R \wedge b : C_1 \wedge \cdots \wedge b : C_k$$

if and only if

$$\langle \mathcal{C}, \mathcal{R}, \mathcal{O} \cup \{b : P_b\} \rangle \models a : \exists R (P_b \sqcap C_1 \sqcap \cdots \sqcap C_k)$$

The theorem directly applies to parts of our query. Consider for example the two conjuncts $(X, Y) : father\text{-}of$ and $Y : Female$. From the theorem we get that the instances of the concept $(\exists father\text{-}of.Female)$ are exactly the objects to instantiate X we were looking for. The introduction of a new primitive concept is not necessary as we are not checking whether a concrete object satisfies the query expressions. The transformation of a complete query is more difficult due to the dependencies between the variables that occur in the query expression. The correct transformation of a query into a concept expression depends on the kinds of dependencies between the variables in the query which is reflected in the structure of the query graph. Using the information provided by the query graph, the transformation of a conjunctive query into a concept expression, called roll-up, can be computed as follows:

Definition 12 (Query Roll-up (Horrocks and Tessaris 2000)). *the roll-up of a query Q with query tree G is a concept expression derived from Q by successively applying the following rule:*

– *If G contains a leaf node y then the role term (x,y):R is rolled up according to theorem 1. The edge (x,y) is removed from G*

Using this roll-up mechanism, we can translate our example query into the concept expression shown in the equation below. The results of Horrocks and Tessaris guarantee that every answer to the query is a member of this concept and vice versa.

$$(\exists father\text{-}of.(Female \sqcap (\exists degree.(\exists awarded\text{-}by.\{vu\})) \sqcap (\exists works\text{-}at.\top)))$$

Using the query graph as a basis for determining the sequence of queries has the advantage that we already obtain the information needed for transforming the query into a concept expression being used to actually answer the query. If we apply the role-up algorithm of Horrocks and Tessaris to the query graphs in figure 2 we get the following sequence of concept expressions defining supersets of the set of answers we are looking for:

Q^1: $\top \sqcap (\exists father\text{-}of.Female)$
Q^2: $\top \sqcap (\exists father\text{-}of.(Female \sqcap (\exists degree.\top) \sqcap (\exists works\text{-}at.\top)))$
Q^3: $\top \sqcap (\exists father\text{-}of.(Female \sqcap (\exists degree.(\exists awarded\text{-}by.\{vu\})) \sqcap (\exists works\text{-}at.\top)))$

In the same way we get the following sequence of concept expressions for the breaths-first strategy:

Q^1: $\top \sqcap (\exists father\text{-}of.Female)$
Q^2: $\top \sqcap (\exists father\text{-}of.(Female \sqcap (\exists degree.\top)))$
Q^3: $\top \sqcap (\exists father\text{-}of.(Female \sqcap (\exists degree.\top) \sqcap (\exists works\text{-}at.\top)))$
Q^4: $\top \sqcap (\exists father\text{-}of.(Female \sqcap (\exists degree.(\exists awarded\text{-}by.\{vu\})) \sqcap (\exists works\text{-}at.\top)))$

For the depths-first strategy, the following sequence is generated:

Q^1: $\top \sqcap (\exists father\text{-}of.Female)$
Q^2: $\top \sqcap (\exists father\text{-}of.(Female \sqcap (\exists degree.\top)))$
Q^3: $\top \sqcap (\exists father\text{-}of.(Female \sqcap (\exists degree.(\exists awarded\text{-}by.\{vu\}))))$

Q^4: $\top \sqcap (\exists father\text{-}of.(Female \sqcap (\exists degree.(\exists awarded\text{-}by.\{vu\})) \sqcap (\exists works\text{-}at.\top)))$

The role of this mechanism is two-fold. As Horrocks and Tessaris show, it can be used to actually execute queries using existing subsumption reasoners. Further, using a description logic reasoner, we can easily prove that the two requirements on the sequence of queries stated in the intriduction of section 4 actually hold. In the cases discussed here, it is straightforward to see that the requirements hold as we are just adding conjuncts ending up with exactly the same query we want to approximate. Using a subsumption reasoner, however, enables us to check the requirements for arbitrary sequences of query.

6 Discussion

Querying terminological knowledge bases is a problem that currently gains a lot of importance as it can be assumed to be one of the main inference tasks on the semantic web. Existing proposals for query answering approaches are able to answer queries over very complex knowledge bases but at the price of a high complexity resulting from the need to perform deductive reasoning over the knowledge base. In order to improve the efficiency of this approach, we propose various strategies for approximating answers by relaxing the query expression to a subsuming one that still returns all correct answers but possibly also some wrong ones. As the main application we have in mind is the search for information on the web, returning some wrong answers is acceptable as the result is returned to the user for validation. Further, our approximation strategies is designed in such a way that the exact result is also computed if there is enough time. These characteristics make our approach a promising one for gaining efficiency in semantic web querying despite the fact that it is very premature. Possible further improvements are the following:

Approximation Heuristics: The different strategies presented in this paper were solely based on the structure of the query graph. The different approximation steps were motivated by uninformed search strategies. However, it has been shown in other domains such as diagnosis or configuration that approximation strategies can be significantly improved using knowledge about the problem domain [12]. In our case, this knowledge is provided in terms of the concept and role definitions in the terminological knowledge base as well as in terms of concepts and roles occurring in the approximated query. An interesting question for further research is whether we can use this domain knowledge to define informed approximation strategies that are not solely guided by the structure of the query graph, but also by the nature of the concepts and roles. In particular, such knowledge can be used to determine the order, in which links are used to expand the query graph. We might want to add those conjuncts first that are most restrictive, thereby giving a better approximation of the actual answer. Such a heuristic ordering could be applied in combination with one of the described strategies or as a completely heuristic strategy that always uses the heuristics to expand the tree.

Integrating Subsumption Reasoning: A second interesting extension of the proposed approach is to use the roll-up mechanism proposed by Horrocks and Tessaris in order to interleave query formulation with subsumption reasoning. The advantage of such an effort is two-fold. First of all, we get a way for checking query subsumption, doing this is straightforward for the approaches we discuss in this paper, but maybe there are more sophisticated strategies that lead to situations where query subsumption is not straightforward. Indeed using a subsumption reasoner to check query subsumption may become an essential part of the query generation process. Secondly, the rolled-up query is the way to actually execute the query on a complex knowledge base and to return results. These results (e.g. the properties of objects in the answer set) may also give hints towards ways to formulate the next query in the sequence. The choice of the next conjunct can then be based on its ability to remove objects from the previous answer set. A possible drawback of interleaving query formulation with subsumption reasoning is the computational complexity of subsumption checking having a negative impact on the overall performance of the approach. This is not so much a problem for the case of query execution as this inference step has to be done anyway. Testing query subsumption however is an additional reasoning task that should be avoided if possible.

Approximate Deduction: Up to now, we only considered approximations that are done in the transformation step of the query answering approach by Horrocks and Tessaris. Another option not being discussed here is to apply approximation techniques to the second step of the approach, namely the deduction step that actually computes the answer to the query based on the concept expression generated by the transformation. Existing approaches to this problem are known as approximate deduction. Examples of techniques for approximate deduction are the work of Cadoli and Schaerf [10] and of Dalal [2]. These techniques for approximate deduction are very general. They are typically dependent on the choice of specific parameters that determine the degree of approximation (e.g. the length of clauses [2] or a subset of the alphabet [10]). In general, it is hard to determine which values for these parameters give the best approximations. Approximate deduction approaches for terminological knowledge, which is relevant for our work have been developed by Selman and Kautz [11] as well as Schaerf and Cadoli [10]. The latter discuss approximations for specific logics that can be used to express the kind of concepts we get by transforming conjunctive queries which makes the work a good starting point for investigating the use of approximate deduction for query answering.

References

1. D. Brickley, R. Guha, and A. Layman. Resource description framework (rdf) schema specification. Working draft, W3C, August 1998. http://www.w3c.org/TR/WD-rdf-schema.
2. Mukesh Dalal. Semantics of an anytime family of reasoners. In *European Conference on Artificial Intelligence*, pages 360–364, 1996.

3. T. Dean and M. Boddy. An analysis of time-dependent planning. In *Proceedings of AAAI-88*, pages 49–54, 1988.
4. Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, , and Andrea Schaerf. Reasoning in description logics. In Gerhard Brewka, editor, *Principles of Knowledge Representation*, Studies in Logic, Language and Information, pages 193–238. CSLI Publications, 1996.
5. D. Fensel. *Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce*. Springer-Verlag, Berlin, 2001.
6. A. Gomez-Perez and O. Corcho. Ontology langauges for the semantic web. *IEEE Intelligent Systems*, January/February:54–60, 2002.
7. Volker Haarslev and Ralf Moller. Description of the RACER system and its applications. In *Proceedings of the Description Logics Worlshop DL-2001*, Stanford, CA, 2001.
8. Alon Y. Halevy. Theory of answering queries using views. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 29(4):40–47, 2000.
9. Ian Horrocks and Sergio Tessaris. A conjunctive query language for description logic aboxes. In *AAAI/IAAI*, pages 399–404, 2000.
10. Marco Schaerf and Marco Cadoli. Tractable reasoning via approximation. *Artificial Intelligence*, 74(2):249–310, 1995.
11. B. Selman and H. Kautz. Knowledge compilation and theory approximation. *Journal of the ACM*, 43(2):193–224, March 1996.
12. A. ten Teije and F. van Harmelen. Exploiting domain knowledge for approximate diagnosis. In M. Pollack, editor, *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI'97)*, pages 454–459, Nagoya, Japan, August 1997.
13. Frank van Harmelen, Peter F. Patel-Schneider, and Ian Horrocks. Reference description of the DAML+OIL (march 2001) ontology markup language. http://www.daml.org/2001/03/reference.html, march 2001.
14. S. Zilberstein and S.J. Russell. Approximate reasoning using anytime algorithms. In S. Natarajan, editor, *Imprecise and Approximate Computation*. Kluwer Academic Publishers, 1995.