Towards Practical Taxonomic Classification for Description Logics on the Semantic Web

Scott Sanner

Department of Computer Science
University of Toronto
Toronto, Ont, M5S 3H5, CANADA
ssanner@cs.toronto.edu

Abstract. Description logics offer a well-defined semantics for many common and useful reasoning tasks that can be formalized under the notion of subsumption. As such, they are currently finding use as a representation language for the Semantic Web (e.g., DAML+OIL) where concepts from distributed knowledge bases can be classified into a taxonomic hierarchy that compactly encodes all subsumption relationships between them. However, there are many practical considerations for designing sound and complete, yet efficient algorithms for performing large-scale taxonomic classification. While structural subsumption algorithms have traditionally provided efficient techniques for performing classification, they have only supported relatively inexpressive languages. Consequently, we propose an approach that extends previous structural subsumption algorithms to support sound and complete classification of a moderately expressive description logic with disjunction. Furthermore, we extend this algorithm to a more expressive language with the claim that its sources of incompleteness are benign in practice. Finally, we show that for a set of assumptions that can be argued to hold for the Semantic Web, both classification algorithms require polynomial time in the size of the original knowledge base. We argue that such a result is a necessity for classification algorithms that will scale with the expected growth of the Semantic Web.

1 Introduction

As languages for knowledge representation on the Semantic Web [1] and other media gain widespread acceptance, it is important that the associated reasoning tools have the ability to scale to the unprecedented quantities of structured content that are likely to emerge. Furthermore, it is important that the majority of this reasoning be automated since direct human involvement is both costly and time-consuming. Consequently, the desire for automated reasoning coupled with the expected growth rate of the Semantic Web pose a set of constraints on reasoning algorithms whose sheer magnitude has not been encountered previously in the field of knowledge representation and reasoning.

1.1 Description Logics and the Semantic Web

As the Semantic Web comes of age, it is likely that there will be a shift from the database-like specification of knowledge bases (KBs) containing primitive concepts (e.g., a list of a retailer's products) to KBs containing logical compositions of concepts collected from distributed KBs. Consequently, the choice of a description logic (e.g., DAML+OIL [2]) as a representation language for the Semantic Web is well-suited to this domain since it automates reasoning about the relationships between such logically composed concepts.

To motivate our discussion we use a simplified example drawn from online retail sales: If three online office supply retailers list their products in DAML+OIL, each could specify the following primitive concept URIs for their pencil products:

$$http://www.ret1.com/pg.daml\#Ret1\_Pencil$$
$$http://www.ret2.com/pg.daml\#Ret2\_Pencil$$
$$http://www.ret3.com/pg.daml\#Ret3\_Pencil$$

It is important to note that each of these concepts is unique and primitive (i.e., lacking definitions of sufficient conditions for membership) and thus have no intrinsic relation to each other outside of natural language interpretation.

Given these primitive concepts, it is expected that other organizations wanting to construct their own product ontologies will define composite concepts referring to the set of pencils that agree with their individual organizational definition and needs. For example:[1]

$$Org1\_Pencil \doteq Ret1\_Pencil \sqcup Ret2\_Pencil$$
$$Org2\_Pencil \doteq Ret1\_Pencil \sqcup Ret2\_Pencil \sqcup Ret3\_Pencil$$
$$Org3\_Pencil \doteq Org1\_Pencil \sqcup Ret3\_Pencil$$

To demonstrate the relationships that can be inferred between the above defined concepts, we have drawn a taxonomy in Figure 1. Arrows represent a subsumption (a.k.a. *kind-of*) relation that points from the subsumed concept to the subsuming concept.

Such an automatically induced taxonomic structure induces a subsumption hierarchy that has two main properties: 1) It encodes all inferrable subsumption relations via the reflexive/transitive closure of subsumption links, and 2) it is minimal in the sense that no subsumption links are redundant. Thus, we can deduce by the mutual subsumption relation between *Org2_Pencil* and *Org3_Pencil* that these concepts are equivalent, and from their subsumer relation to *Org1_Pencil* that both are more general than this concept. Note that while these relationships can be provably derived from the concept definitions, all relationships are not immediately apparent from the definitions.

Consequently, it is the purpose of this paper to describe an efficient and practical algorithm for taxonomic classification of concepts composed from a set

[1] See Table 1 for the definition of the description logic symbols used in these descriptions.

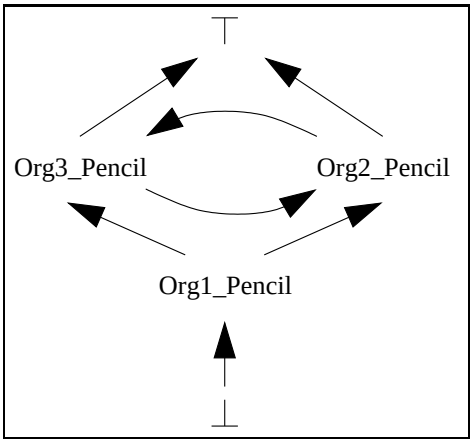

Fig. 1. A taxonomy of defined concepts.

of logical constructors.[2] Fortunately, such reasoning has been extensively studied in the field of description logics.

1.2 Theoretic and Empirical Preliminaries

Theoretic research in description logics has provided worst-case computational complexity bounds for sound and complete subsumption inference under a model-theoretic semantics for a variety of description languages [3]. These results show that reasoning is coNP-hard or worse with respect to the combined length of the concepts for even relatively inexpressive languages (e.g., $\{C \sqcap D, \exists R, \forall R.C, R|_C\}$ [4]).

Unfortunately, this is only the beginning of the intractability issues for subsumption. If one considers a terminological formalism where concepts can be defined in terms of other concepts (a necessity for the Semantic Web reasoning framework previously outlined), then as pointed out by Nebel [5], reasoning is intractable for even very simple languages (i.e., $\{C \sqcap D, \forall R.C, \doteq\}$).

However, these results on computational complexity do not appear to doom all expressive concept and term subsumption to the realm of intractability since many researchers have argued that the worst-case subsumptions tend to occur rarely in practice [6, 7]. And empirically, recent work on complete satisfiability-based subsumption for expressive terminological languages has yielded algorithms that perform quite efficiently on many real-world knowledge bases [8, 9].

[2] It is important to note here that we are *not* trying to solve the data integration problem. Rather we are examining contexts for the use of description logic classification on the Semantic Web and designing practical algorithms to support such reasoning.

1.3 Problems with Current Approaches

While a lot of recent description logic research has focused on practically efficient algorithms, most of this work has focused on the tractability of an individual subsumption test. This approach has been criticized by Woods [10], who states that the "primary tractability concern is not the cost of subsumption, but the cost of classifying into a large taxonomy". That is, *the cost of individual subsumption testing does not grow with the size of the knowledge base but the cost of classification clearly does.* And this criticism is especially relevant given the large quantities of content that will need to be classified on the Semantic Web.

As pointed out by both Nebel [5] and Woods [10], structural subsumption algorithms lend themselves to efficient techniques for terminological classification by caching redundant subsumption computations. Unfortunately, while the use of structural subsumption algorithms has yielded provably efficient classification algorithms (e.g., Classic [11]), it has traditionally been limited due to its inability to provide complete reasoning for many commonly used combinations of description logic constructors [12].

1.4 Constraints for the Semantic Web

Before we provide a potential solution to description logic classification on the Semantic web, let us first define the constraints on language expressiveness and reasoning complexity posed by this domain:

1. *The taxonomic classification algorithm should run in polynomial-time in the size of the knowledge base.* We assume any time complexity beyond polynomial to be intractable when applied to the Semantic Web. While it has been frequently argued that only linear or sublinear algorithms would be considered tractable for Semantic Web reasoning, it is inconceivable that a reasonably complete, linear-time classification algorithm could be provided for the languages discussed here.
2. *The language should have adequate expressiveness for building practically useful concepts and for reasoning over distributed KBs.* We would like to have conjunction and existential, universal, and qualified cardinality role restrictions since these are commonly used constructors for forming defined concepts. And we would also like disjunction since it is one of the primary mechanisms for building composite concepts from distributed KBs.

1.5 Towards a Potential Solution

With these requirements for reasoning on the Semantic Web, it seems we are at a bit of an impasse: The approaches to subsumption for expressive languages (e.g., satisfiability checking) do not place their primary focus on efficient taxonomic classification and the structural approaches do not seem to support expressive enough languages for our purposes. However, given that structural subsumption techniques are known to lend themselves to efficient classification algorithms, it may be advantageous to explore extensions to these algorithms.

In surveying some of the major research efforts in structural subsumption [13, 14, 6, 15, 10, 11], it becomes apparent that no work has provided a provably complete algorithm for structural subsumption in either 1) a language that uses both conjunction and disjunction, or 2) a language that uses both both universal and qualified existential role restrictions. Consequently, any structural classification algorithm intended to satisfy the above constraints will either have to close these completeness gaps while maintaining polynomial complexity for applications on the Semantic Web, or argue that such incompleteness is benign for applications in this domain. We will use both of these approaches and in doing so will adopt the stance of many researchers who have argued that such practical considerations should be the primary goal of reasoning algorithms in light of the need for expressive languages and their inherent intractability results [16, 5, 10].

2 Language and Semantics

Let us now define the language elements and semantics for a language containing the constructors that we have previously argued will be required for reasoning on the Semantic Web.

2.1 Language Definition

Table 1 lists the constructor, definitional, and axiomatic elements for two languages, $\mathcal{L}_1$ and $\mathcal{L}_2$.[3] Both languages are a subset of the DAML+OIL language used for the Semantic Web.[4] We consider $\mathcal{L}_1$ and $\mathcal{L}_2$ separately since the introduction of universal and max cardinality role restrictions in $\mathcal{L}_2$ introduces incompleteness for our structural subsumption algorithm.

2.2 Additional Language Restrictions

Since we will be using structural subsumption for purposes of efficient classification, we must make a few additional restrictions on the language to make it amenable to this approach.

1. We must prevent any non-primitive concept from referencing itself or a subclass in a definition (including a reference made through a role restriction). This restriction results in an acyclic terminology.
2. We limit axioms to primitive concepts only. However this still allows us to define axioms of the form $C \sqsubseteq D \sqcap E$ or $D \sqcup E \sqsubseteq C$ since they can be inferred from axioms between primitive concepts.
3. We limit non-primitive concepts to have at most one definition.

[3] In the DL naming scheme, $\mathcal{L}_2$ is *similar* to $\mathcal{FL}^{-}\mathcal{ENUQH}$, but given the additional language restrictions required by the structural approach, it seems best to stick with $\mathcal{L}_2$ to avoid confusion of language properties.

[4] DAML+OIL itself is a slight extension of the $\mathcal{SHIQ}$ DL.

Languages $\mathcal{L}_1$ and $\mathcal{L}_2$

Constructor	*Syntax*	*Semantics*
Concept name	C	$C^{\mathcal{I}}$ (where $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$)
Top (Thing)	$\top$	$\Delta^{\mathcal{I}}$
Bottom (Nothing)	$\bot$	$\emptyset$
Conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
Disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
Existential restriction	$\exists R.C$	$\{x \mid \exists y : R^{\mathcal{I}}(x,y) \wedge C^{\mathcal{I}}\}$
Min cardinality restriction	$\geqslant nR$	$\{x \mid \sharp\{y \vert R^{\mathcal{I}}(x,y)\} \geq n\}$
Qualified min cardinality restriction	$\geqslant nR.C$	$\{x \mid \sharp\{y \vert R^{\mathcal{I}}(x,y) \wedge C^{\mathcal{I}}(y)\} \geq n\}$
Role name	R	$R^{\mathcal{I}}$ (where $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$)

Definitional or Axiomatic Constraint	*Syntax*	*Semantic Constraint*
Concept definition	$C \doteq D$	$C^{\mathcal{I}} \equiv D^{\mathcal{I}}$
Concept subsumption axiom	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
Role subsumption axiom	$R \sqsubseteq S$	$R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$

Language $\mathcal{L}_2$ only

Constructor	*Syntax*	*Semantics*
Universal (value) restriction	$\forall R.C$	$\{x \mid \forall y : R^{\mathcal{I}}(x,y) \rightarrow C^{\mathcal{I}}\}$
Max cardinality restriction	$\leqslant nR$	$\{x \mid \sharp\{y \vert R^{\mathcal{I}}(x,y)\} \leq n\}$
Qualified max cardinality restriction	$\leqslant nR.C$	$\{x \mid \sharp\{y \vert R^{\mathcal{I}}(x,y) \wedge C^{\mathcal{I}}(y)\} \leq n\}$

Table 1. Syntax and semantics for $\mathcal{L}_1$ and $\mathcal{L}_2$

2.3 Semantics

We provide a model-theoretic semantics for $\mathcal{L}_1$ and $\mathcal{L}_2$ in Table 1. Under an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, $\Delta^{\mathcal{I}}$ is a nonempty domain, and an interpretation function $\cdot^{\mathcal{I}}$ maps from concept names into a subset of $\Delta^{\mathcal{I}}$ and from role names into a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.

2.4 Subsumption

Under a model-theoretic semantics, we have the following definition of subsumption with respect to a KB Σ: $\Sigma \models C \sqsubseteq D$ *iff* $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ *in every model* $\mathcal{I}$ *of* Σ.

Structural subsumption is a procedural method for determining subsumption with respect to a KB by using structural comparison rules and it is denoted here by $\sqsubseteq_s$.[5] Such rules for subsumption in $\mathcal{L}_1$ and $\mathcal{L}_2$ are given in Table 2.

[5] If the structural comparison rules are sound and complete with respect to the model-theoretic semantics, then $\sqsubseteq$ and $\sqsubseteq_s$ are equivalent relations.

Structural comparison of two normalized concept structures

Concept A type	*Concept B type*	*Structural comparison rule for* $A \sqsubseteq_s B$
Primitive	Primitive	(Base) $A \sqsubseteq_s B$ iff $A \sqsubseteq_a^* B$.
Primitive	Conjunctive	(Recursive) $A \sqsubseteq_s B$ iff for *every* B_j, $A \sqsubseteq_s B_j$.
Primitive	Disjunctive	(Recursive) $A \sqsubseteq_s B$ iff for *some* B_j, $A \sqsubseteq_s B_j$.
Conjunctive	Primitive	(Recursive) $A \sqsubseteq_s B$ iff for *some* A_i $A_i \sqsubseteq_s B$.
Conjunctive	Conjunctive	(Recursive) $A \sqsubseteq_s B$ iff for *every* B_j, there is *some* A_i such that $A_i \sqsubseteq_s B_j$.
Conjunctive	Disjunctive	(Recursive) $A \sqsubseteq_s B$ iff for *some* A_i and *some* B_j, $A_i \sqsubseteq_s B_j$.
Disjunctive	Primitive	(Recursive) $A \sqsubseteq_s B$ iff for *every* A_i, $A_i \sqsubseteq_s B$.
Disjunctive	Conjunctive	(Recursive) $A \sqsubseteq_s B$ iff for *every* A_i and *every* B_j, $A_i \sqsubseteq_s B_j$.
Disjunctive	Disjunctive	(Recursive) $A \sqsubseteq_s B$ iff for *every* A_i, there is *some* B_j such that $A_i \sqsubseteq_s B_j$.

Structural comparison of two normalized role restriction structures

Restriction A type	*Restriction B type*	*Structural comparison rule for* $A \sqsubseteq_s B$
$\leqslant n_1 R_1.C_1$	$\leqslant n_2 R_2.C_2$	(Recursive) $A \sqsubseteq_s B$ iff $n_1 \leq n_2$ and $R_1 \sqsubseteq_r^* R_2$ and $C_1 \sqsubseteq_s C_2$.
$\geqslant n_1 R_1.C_1$	$\geqslant n_2 R_2.C_2$	(Recursive) $A \sqsubseteq_s B$ iff $n_1 \geq n_2$ and $R_1 \sqsubseteq_r^* R_2$ and $C_1 \sqsubseteq_s C_2$.
$\forall R_1.C_1$	$\forall R_2.C_2$	(Recursive) $A \sqsubseteq_s B$ iff $R_1 \sqsubseteq_r^* R_2$ and $C_1 \sqsubseteq_s C_2$.
Any other combination		(Base) $A \sqsubseteq_s B$ not possible, return *false*.

Note 1: A_i and B_j indicate constituents of their respective structured concepts.
Note 2: $\sqsubseteq_s$ indicates structural subsumption.
Note 3: $\sqsubseteq_a^*$ indicates the reflexive/transitive closure of concept subsumption axioms.
Note 4: $\sqsubseteq_r^*$ indicates the reflexive/transitive closure of role subsumption axioms.

Table 2. Structural subsumption rules for $\mathcal{L}_1$ and $\mathcal{L}_2$

However it should be noted that applying these subsumption rules directly to any two concept structures would *not* yield model-theoretic completeness. For model-theoretic completeness (at least in $\mathcal{L}_1$), a concept's structure must first be normalized before these rules are applied – this will be discussed in section 3.

2.5 Inherent Complexity of Subsumption for $\mathcal{L}_1$ and $\mathcal{L}_2$

There are no known complexity results for concept or term subsumption in $\mathcal{L}_1$.[6] Consequently, although we will present a complete, polynomial-time subsumption algorithm for normalized concepts in $\mathcal{L}_1$, it is unclear whether a polynomial-time algorithm also exists for unnormalized concepts in $\mathcal{L}_1$. However, while we

[6] However, we note that if proving intractability for $\mathcal{L}_1$ were possible, it would require some ingenuity since there is no obvious way to simulate negation as required for reductions from many standard NP-complete problems.

will not be able to claim a worst-case polynomial-time algorithm for subsumption in $\mathcal{L}_1$, it does appear that we have avoided some of the more problematic constructor combinations known to exist in description logics (e.g., the $\forall R.C$ constructor that was required by Nebel [5] to show that terminological reasoning is intractable).

Unfortunately, $\mathcal{L}_2$ is a reasonably expressive language for which many known intractability results have been proved. A small subset of it (i.e., $\{\sqcap, \exists R.C, \forall R.C, \leqslant nR\}$, commonly known as $\mathcal{FL}^-\mathcal{EN}$) has been shown to be coNP-hard for subsumption [3]. And another subset (just $\{\sqcap, \forall R.C, \doteq\}$) has been shown to be intractable for terminological subsumption [5]. Nevertheless, there are many arguments as to why these sources of intractability either do not occur in practice or can be ignored altogether – these will be discussed in section 4.

3 Algorithm Definition

Once the taxonomy has been initialized, the classification algorithm seeks to classify a new concept into the taxonomy according to the structural subsumption rules such that the reflexive/transitive closure of all taxonomy links yields all subsumption relationships (w.r.t. this definition) while ensuring that no links are redundant (i.e., the taxonomy is minimal). Due to space restrictions, we only present a general overview of the classification algorithm here. However, the interested reader should consult the technical report [17] for a code-level presentation of the algorithm along with a discussion of implementation details and extensions to relax some of the restrictions stated here.[7]

At certain points in the following discussion, we will make reference to Figure 2 which represents a completely classified taxonomy of concepts. Note that this taxonomy makes important distinctions among subsumption links that will be used to enhance the efficiency of the algorithm. The a, c, d, and s links respectively stand for axiomatic, conjunctive definition, disjunctive definition, and structural subsumption.

3.1 Initialization of the Knowledge Base

We begin classification of a knowledge base by initializing it; This simply requires that we insert the root concepts $\top$ and $\bot$, the root role $\top_{role}$, all roles and primitive concepts, and any stated and default axioms relating them (using a links as in Figure 2).

3.2 Classification of a Concept

Next we classify each individual structured concept until all concepts are classified in the taxonomy. Due to the presence of mutual recursion between the

[7] This report also discusses a freely available implementation of the general algorithm described here.

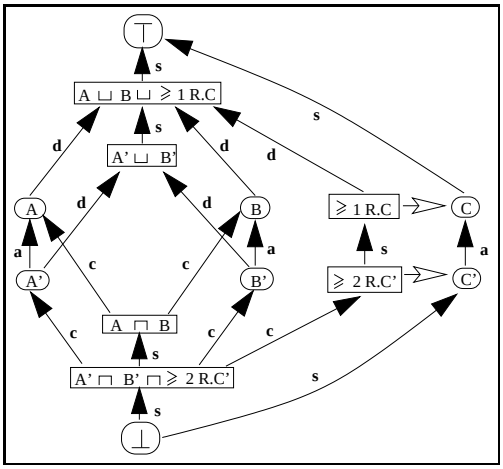

Fig. 2. Example taxonomy showing different subsumption link types (closed arrow) as well as concept referents of restrictions (open arrow). The a, c, d, and s labels respectively stand for axiomatic, conjunctive definition, disjunctive definition, and structural subsumption links. Note the rules from Table 2 used to determine subsumption here.

normalization and classification algorithms, the algorithm can appear somewhat complex at first. Consequently, we will first discuss the following general steps for performing taxonomic classification of a concept:

1. *Expand and normalize the concept description*: Expand all concept term definitions.[8] Next, apply a normalization procedure to ensure model-theoretic completeness of structural comparison.
2. *Find the most specific subsumers (MSS)*: Given a target concept, find its set of most specific subsumers (mss) in the taxonomy.
3. *Find the most general subsumees (MGS)*: Given a target concept, find its set of most general subsumees (mgs) in the taxonomy.
4. *Insert the concept into taxonomy*: Given a target concept and its mss and mgs sets, modify the link structure to incorporate the newly inferred subsumptions while maintaining link minimality.

In the following sections, we elaborate on each of the above steps in more detail.

Expansion and Normalization Algorithm Expansion and normalization are required to ensure that structural comparison of two defined concepts will yield all subsumption relationships with respect to the model-theoretic semantics. This procedure follows:

[8] It is actually the case that concepts referenced by existential and min cardinality role restrictions do not need to be expanded in order for structural comparison to be complete. However, we omit this optimization since it significantly simplifies our algorithm presentation and theoretic discussion.

1. *Expand definitions*: Replace any non-primitive concept by its definition until only primitive concepts remain.
2. *Convert to DNF*: Convert the concept to disjunctive normal form.[9]
3. *Convert restrictions to canonical form*: For each role of the following type, perform the given syntactic rewrite:
   a) $\exists R.C \longrightarrow \geqslant 1R.C$
   b) $\geqslant nR \longrightarrow \geqslant nR.\top$
   c) $\leqslant nR \longrightarrow \leqslant nR.\top$
4. *Merge conjoined universal role restrictions*: If any universal role restrictions in the same conjunction share the same role, merge them and conjoin their referent concepts.
5. *Classify all role restrictions*: Recursively classify all role restrictions.
6. *Rewrite tautologous and inconsistent role restrictions*: Rewrite all role restrictions determined to be equivalent to the following forms:
   a) $\forall R.\top \longrightarrow \top$
   b) $\leqslant nR.\bot \longrightarrow \top$
   c) $\geqslant 0R.C \longrightarrow \top$
   d) $\geqslant nR.\bot$ where $n \geq 1 \longrightarrow \bot$
7. *Eliminate redundant and inconsistent conjoined elements*: Remove constituents that subsume other constituents within a conjunction. Also, if two restrictions $\leqslant n_1R_1.C_1$ and $\geqslant n_2R_2.C_2$ exist within a conjunction and $R_1 \sqsubseteq R_2$ and $C_1 \sqsubseteq C_2$ and $n_1 > n_2$, then replace the entire conjunction with $\bot$.
8. *Merge, classify, rewrite, and eliminate disjoined universal role restrictions*: If two different disjoined sets of conjunctions are equivalent in every way except for a single universal role restriction in each which agree on their restriction type and relation, then disjoin the referent concepts in these restrictions and merge the two conjunctions. Next, classify the newly disjoined restriction referent and rewrite it according to step 6. Next, eliminate the restriction if it is redundant with any other conjoined elements as in step 7. Finally, repeat these steps until no further disjunctive role merges are possible.
9. *Build and classify conjunctive substructure*: Since a DNF definition is broken down into two levels (i.e., a disjunction of conjunctions) build concepts for each of the conjunctions, link these concepts to their constituents (c links in Figure 2), and recursively classify them.
10. *Eliminate redundant and tautologous disjoined elements*: Remove constituents that are subsumed by other constituents within a disjunction. Also, if two restrictions $\leqslant n_1R_1.C_1$ and $\geqslant n_2R_2.C_2$ exist standalone within a disjunction and $R_2 \sqsubseteq R_1$ and $C_2 \sqsubseteq C_1$ and $n_1 \leq n_2 + 1$, then replace the entire disjunction with $\top$.
11. *Build disjunctive concept structure*: Now that we have the conjunctive components of the DNF definition built and classified, disjunctively link this concept to each of its constituents (d links in Figure 2). Once this is complete, the target concept is ready for classification.

[9] This step is required for completeness of the structural rule comparing a potential conjunctive subsumee with a potential disjunctive subsumer. See the technical report [17] for more details.

Most Specific Subsumer (MSS) As noted in the above explanation of taxonomic classification, only defined concepts (i.e., those with conjunctive or disjunctive definitions) and role restrictions are passed to the classification algorithm. Consequently, we define an MSS algorithm to handle these respective concept structures:

- *Conjunctively defined target*: If we are looking for the *conjunctive MSSs* of a conjunctively defined concept, we can do this in two steps.
  1. We mark the target and all parents in the taxonomy up to $\top$ with a distinct mark (following all a, c, and s links).
  2. Now, starting with a search frontier containing the $\top$ concept, we search down the taxonomy from all elements in this frontier (following all a, c, and s links), adding any marked concept or any conjunctively defined concept that has all constituents (i.e., c links) marked. If during this search, any concept is a subsumer but has no children that are subsumers, this concept is placed in the candidate mss set.

  Note that, according to the subsumption rules in Table 2, we do not have to search for *primitive or disjunctive MSSs* of a conjunctive concept since these subsumptions would automatically be implied by the transitive closure of concept subsumption axioms.
- *Disjunctively defined target*: If we are looking for the MSSs of a disjunctively defined concept, we must check for a few different subsumption cases:
  1. *Disjunctive and primitive MSSs*: We mark up from each of the target's constituents (accessed via d links) all the way to $\top$ with a distinct mark corresponding to the constituent (During the marking up process from constituents, we follow *all* taxonomy links). Then we collect all parent concepts of *one* of the constituents that contain all of the distinct marks and put these in the candidate mss set.
  2. *Conjunctive MSSs*: For the special case of a conjunctive subsumer, we find those that would be MSSs by simply checking to see if any conjunctively defined child of a concept in the candidate mss set has all constituents (i.e., c linked concepts) within this set; If so, we add it to the candidate mss set.
- *Restriction target*: There are a number of ways to find parent subsumers of restrictions and here we present a somewhat inefficient but relatively simple method for doing this. We assume that all concepts have been classified in a complete taxonomy so that finding a subsuming restriction according to the structural subsumption rules simply consists of two steps:
  1. All restrictions making reference to the restriction referent or one of its parents is collected in a set.
  2. All restrictions in this set are checked against the structural subsumption rules to prune out the non-subsumers. This remaining set is the candidate mss set.
- As a final step during MSS, it is important to filter the candidate mss set to remove subsumers that are not most specific.

Most General Subsumee (MGS) Just as in the above MSS algorithm, we define the MGS algorithm for conjunctively and disjunctively defined concepts and role restrictions. One of the elegant symmetries in this algorithm is that it essentially mirrors the MSS algorithm in that it infers from the $\bot$ concept upward (rather than $\top$ downward) and the inference algorithms for conjunction and disjunction are effectively swapped. This similarity is no coincidence and reflects the symmetry of conjunctive and disjunctive subsumption for the structural subsumption definition.

Since this algorithm mirrors the MSS algorithm so closely, we will only provide a brief discussion outlining the differences:

- *Conjunctively defined target*: The algorithm for finding conjunctive MGSs is essentially the same algorithm used to find disjunctive MSSs except that now we pass a distinct mark down from each of the conjunctive constituents and collect all MGSs below *one* of the constituents. This finds all potential conjunctive and primitive MGSs; An additional subsumption check for disjunctive MGSs is performed by adding any disjunctive parents of concepts in the candidate *mgs* set that have all constituents within this set.
- *Disjunctively defined target*: This algorithm is essentially the same as conjunctive MSS except that we mark down from target to $\bot$, initiate our search at $\bot$, and expand our search frontier upward.
- *Restriction target*: This algorithm is essentially identical to the MSS algorithm for restrictions except that the direction of inference is reversed.
- And finally, as in the MSS algorithm, we filter the candidate *mgs* set to remove subsumees that are not most general.

Link Maintenance Once we have determined the *mss* and *mgs* sets for a concept, we add new structural subsumption links (s links) from the target to each element of these sets and remove all s links between the *mss* and *mgs* sets (which are now redundant). Figure 2 shows an example minimal taxonomy (minus the role hierarchy) for a fully classified KB.

3.3 Efficiency Remarks

Perhaps the single most important feature of this algorithm that differentiates it from most, if not all, previous work in description logic classification is that it does not define the classification procedure in terms of independent calls to a subsumption test. Rather, it defines classification as a search procedure that makes at most *one pass*[10] through the taxonomy in order to find all potential MSSs and MGSs that agree with the structural subsumption rules.

The cost savings of this search-based approach are considerable as the taxonomy size grows: Whereas any classification algorithms defined in terms of subsumption tests can potentially make an entire pass through the taxonomy

[10] By one pass we mean that the procedure visits every concept and link in the taxonomy at most once.

for *every* test required during classification (and the number of subsumption tests clearly grows with the size of the taxonomy), this algorithm effectively determines all subsumptions with a *single* pass; It does so by ordering search in a manner that allows all work done for previous subsumption tests to be leveraged for subsequent subsumption tests. Consequently, this refinement offers a remarkable efficiency improvement over previous classification approaches.

4 Theoretic Results and Practical Issues

Due to space limitations, we are unable to state full proofs for the following theorems. Therefore we summarize the theoretical results and refer the interested reader to the technical report [17].

4.1 Soundness and Completeness

Theorem 1. *The classification algorithm is sound with respect to the model theoretic semantics for $\mathcal{L}_1$ and $\mathcal{L}_2$ given in Table 1.*

Proof Sketch. It is straightforward to prove that the classification algorithm correctly determines subsumption with respect to the structural subsumption rules in Table 2. Since these rules can be proved sound via structural induction, the classification algorithm can also be proved sound. □

Theorem 2. *The classification algorithm is complete with respect to the model theoretic semantics for $\mathcal{L}_1$ given in Table 1.*

Proof Sketch. Fortunately, the algorithm simplifies considerably in the absence of the $\mathcal{L}_2$ constructors and thus it is relatively straightforward to show that normalization preserves the extension of a concept. In addition, one can show by structural induction on *normalized* concept structures that the subsumption rules in Table 2 are complete. From this and the previously proved correctness of the algorithm w.r.t. the structural subsumption rules, the classification algorithm can be proved complete for $\mathcal{L}_1$. □

Theorem 3. *The classification algorithm is incomplete with respect to the model theoretic semantics for $\mathcal{L}_2$ given in Table 1.*

Proof. We can trivially prove incompleteness by providing a counterexample. The algorithm does not infer the following subsumption using the classification algorithm although it holds with respect to the model-theoretic semantics:
$(\forall R.C_1 \sqcap \exists R.C_2) \sqsubseteq \exists R.C_1$ □

It is interesting to note that the above incompleteness (and in fact all incompleteness) in $\mathcal{L}_2$ arises from the interaction between different role restriction types.[11] However, we will argue that such incompleteness is relatively benign for

[11] We obviously don't have this problem in $\mathcal{L}_1$ where all role restrictions can be normalized to min cardinality.

the Semantic Web since it is our experience that 1) concepts are typically defined with orthogonal role restrictions[12], and 2) the cases that are not orthogonal can typically be handled via normalization. One can always manufacture concept structures for which incompleteness cannot be resolved easily via normalization, but we would expect such structures to occur rarely, if at all, in practice. Consequently, we would expect the relative frequency of missed subsumptions due to these uncaught interactions to be reasonably small and therefore benign.

4.2 Time and Space Complexity

We define the *size of a concept* as a constant times the length of its syntactic representation. And we define the *size of a taxonomy* as the space required to store the normalized concept structures and the set of taxonomic subsumption links between them. For both, we will denote the *size function* as $s(\cdot)$.

Theorem 4. *Given a normalized concept c_n and a taxonomy t, the time required to classify c_n in t using the algorithm presented here is $O(s^2(c_n) \cdot s(t))$.*

Proof Sketch. Once a concept has been normalized and inserted into the taxonomy, there are two costs associated with classification: 1) An amortized analysis shows that each classification can incur at most $O(s(t) + s(c_n) \cdot s(t))$ operations. 2) And in the worst case, we may need to perform $O(s(c_n))$ sub-classifications of role restriction referents and/or anonymous conjunctive concepts. Consequently, we can bound the number of operations for a classification by $O(s^2(c_n) \cdot s(t))$. □

Hypothesis 1. *Normalization of a concept in $\mathcal{L}_1$ or $\mathcal{L}_2$ requires space polynomial in its original size for the expected class of concept structures on the Semantic Web.*

Discussion. An analysis of the normalization algorithm will show that there are only two sources of super-polynomial expansion during normalization. The first source is from expanding terms with their full definition and we will defer to Nebel's arguments [6, 5] for the case that super-polynomial term expansions are extremely atypical in practice. The second source is from DNF conversion. We defer to the technical report [17] for an in-depth argument of polynomial-space DNF conversion for Semantic Web concepts but we will briefly outline that argument here: The basic idea is that it is natural to build terms by first generating subterms of the correct specificity (using conjunction), and then disjoining them as a means of generalization.[13] In doing so, the natural structure of concepts on the Semantic Web is expected to be very close to DNF (i.e., global disjunctions of local conjunctions) and therefore the sources of exponential blowup in DNF expansion (i.e., conjunctions of disjunctions) are unlikely to occur. □

[12] By orthogonal, we mean that the restrictions do not mutually constrain the same role. Some common exceptions to this rule are already handled by the normalization algorithm (e.g., potential conflicts between min and max cardinality constraints).

[13] Note that this is precisely the paradigm for distributed KB generalization in the introductory example.

Theorem 5. *If Hypothesis 1 holds for a knowledge base in $\mathcal{L}_1$ or $\mathcal{L}_2$, then classification of a KB requires time polynomial in the size of the original KB.*[14]

Proof Sketch. Under Hypothesis 1, the taxonomy is populated with polynomial expansions of original KB concepts, all operations required for classification are polynomial-time in the taxonomy size, and taxonomy growth can be shown to be bounded quadratically. Thus, classification of a KB can be shown to require polynomial-time in the cumulative size of its original unnormalized concepts. □

5 Conclusion

5.1 Summary

The previous decade of description logic research has seen little focus on structural subsumption techniques likely due to the perception that it is unable to handle expressive languages in a sound and complete manner. However, we intended to challenge this position by 1) providing novel extensions to previous algorithms, and 2) showing that if one is willing to make tradeoffs for application specificity, then expected-case polynomial-time, sound, and complete (or incomplete, but aguably benign) classification algorithms for reasonably expressive description languages can be a reality. Furthermore, such tradeoffs almost seem a necessity for reasoning on the scale of the Semantic Web.

5.2 Future Work

There are at least two important questions to be answered by future work in this domain: One question is how to handle additional expressiveness using a structural subsumption definition and associated classification algorithm. Could this approach be extended to languages including complement, transitive roles, or an assertional formalism? Another question is how to optimize the efficiency of these classification algorithms. For example, could classification search be pruned based on provable constraint techniques? Or, could concepts be classified in an optimal order that reduces redundant work in the MSS or MGS procedures? At the very least, future work along both of these lines will be a necessity to ensure that description logic classification algorithms can practically apply and tractably scale to the many applications of description logic reasoning that will likely emerge as the Semantic Web matures.

Acknowledgements

The author would like to thank Bill Woods for providing the fundamental insights, motivations, and foundational work that underlie many of the key ideas in this paper. And the author would also like to thank the Stanford University Knowledge Systems Lab for providing funding for the majority of this work.

[14] Note that we do not give a specific polynomial bound for classification. If one could empirically argue for a specific polynomial bound on normalization expansion (e.g., linear), then one could derive a corresponding specific bound on KB classification.

References

1. Bemers-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American **284** (2001) 34–43
2. Horrocks, I., van Harmelen, F., Patel-Schneider, P.: DAML+OIL language specification (2001) Document located on-line at http://www.daml.org/2001/03/daml+oil-index.html.
3. Donini, F.M.: Complexity of reasoning. In Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F., eds.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2002)
4. Levesque, H.J., Brachman, R.J.: A fundamental tradeoff in knowledge representation and reasoning (revised version). In Brachman, R.J., Levesque, H.J., eds.: Readings in Knowledge Representation. Morgan-Kaufmann, Inc. (1985) 41–70
5. Nebel, B.: Terminological reasoning is inherently intractable. Artificial Intelligence **43** (1990) 235–249
6. Nebel, B.: Reasoning and Revision in Hybrid Representation Systems. Lecture Notes in Artificial Inteliigence 422. Springer-Verlag, Berlin, Heidelberg, New York (1990)
7. Horrocks, I., Patel-Schneider, P.: Optimising description logic subsumption. Journal of Logic and Computation **9** (1999) 267–293
8. Horrocks, I.R.: Using an expressive description logic: FaCT or fiction? In Cohn, A.G., Schubert, L., Shapiro, S.C., eds.: KR'98: Principles of Knowledge Representation and Reasoning. Morgan Kaufmann, San Francisco, California (1998) 636–645
9. Horrocks, I., Sattler, U., Tobies, S.: Practical reasoning for very expressive description logics. Logic Journal of the IGPL **8** (2000) 239–264
10. Woods, W.A.: Understanding subsumption and taxonomy: A framework for progress. In Sowa, J., ed.: Principles of Semantic Networks: Explorations in the Representation of Knowledge. Morgan Kaufmann, San Mateo, US (1991)
11. Borgida, A., Patel-Schneider, P.F.: A semantics and complete algorithm for subsumption in the CLASSIC description logic. Journal of Artificial Intelligence Research **1** (1994) 277–308
12. Brachman, R.J., McGuiness, D.L., Patel-Schneider, P.F., Resnick, L.A.: Living with CLASSIC: when and how to use a KL-ONE-like language. In Sowa, J., ed.: Principles of Semantic Networks: Explorations in the Representation of Knowledge. Morgan Kaufmann, San Mateo, US (1991)
13. Nebel, B.: Computational complexity of terminological reasoning in BACK. Artificial Intelligence **34** (1988) 371–383
14. Patel-Schneider, P.F.: Undecidability of subsumption in nikl. Artificial Intelligence **39** (1989) 263–272
15. MacGregor, R.: The evolving technology of classification-based knowledge representation systems. In Sowa, J., ed.: Principles of Semantic Networks: Explorations in the Representation of Knowledge. Morgan-Kaufmann, Inc. (1991) 385–400
16. Doyle, J., Patil, R.: Two theses of knowledge representation: Language restrictions, taxonomic classifications, and the utility of representation services. Artificial Intelligence **48** (1991) 261–298
17. Sanner, S.P.: Towards practical taxonomic classification for description logics on the semantic web. Technical Report KSL-03-06, Stanford University Knowledge Systems Lab (2003) Located on-line at http://www.ksl.stanford.edu.