WSReasoner: A Prototype Hybrid Reasoner for $\mathcal{ALCHOI}$ Ontology Classification using a Weakening and Strengthening Approach

Weihong Song[1], Bruce Spencer[1,2], and Weichang Du[1]

[1] Faculty of Computer Science, University of New Brunswick, Fredericton, Canada
{song.weihong, bspencer, wdu}@unb.ca,
[2] National Research Council, Canada

Abstract. In the ontology classification task, consequence-based reasoners are typically significantly faster while tableau-based reasoners can process more expressive DL languages. However, both of them have difficulty to classify some available large and complex $\mathcal{ALCHOI}$ ontologies with complete results in acceptable time. We present a prototype hybrid reasoning system WSReasoner, which is built upon and takes advantages of both types of reasoners to provide efficient classification service. In our proposed approach, we approximate the target ontology $\mathcal{O}$ by a weakened version $\mathcal{O}_{wk}$ and a strengthened version $\mathcal{O}_{str}$, both are in a less expressive DL $\mathcal{ALCH}$ and classified by a consequence-based main reasoner. Classification of $\mathcal{O}_{wk}$ produces a subset of subsumptions of ontology $\mathcal{O}$ and the target of the classification of $\mathcal{O}_{str}$ is to produce a superset of subsumptions of $\mathcal{O}$. Additional subsumptions derived from $\mathcal{O}_{str}$ may be unsound, so they are further verified by a tableau-based assistant reasoner. For the $\mathcal{ALCHOI}$ ontologies in our experiment, except for one for which WSReasoner has not obtained the result, (1) the number of subsumptions derived from WSReasoner is no fewer than from the reasoners that could finish the classification; (2) WSReasoner takes less time than tableau-based reasoners when the $\mathcal{ALCHOI}$ ontologies are large and complex.

1 Introduction

Ontology classification — computing the subsumption relationships between classes — is one of the foundational reasoning tasks provided by many reasoners. Tableau-based and consequence-based reasoners are two dominant types of reasoners that provide the ontology classification service. Tableau-based reasoners, such as HermiT [8], Fact++ [13] and Pellet [12], try to build counter-models $A \sqcap \neg B$ for candidate subsumption relations, based on sound and complete calculi such as [4] and [8]. These reasoners are able to classify ontologies in expressive DLs like $\mathcal{SROIQ}(\mathcal{D})$.

Consequence-based reasoners classify the ontology based on specifically designed inference rules for deriving logical consequences of the axioms in the ontology. Initially developed for the family of tractable DLs like $\mathcal{EL}^{++}$ [1], these procedures were later extended to Horn-$\mathcal{SHIQ}$ [5] and $\mathcal{ALCH}$ [11] while preserving optimal computational complexity. Reasoners belonging to this category, such as CB, ConDOR, ELK [6], CEL [1] and TrOWL [9], are usually very fast and use less memory.

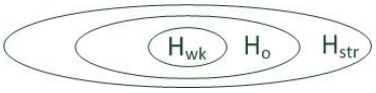

Fig. 1. Subsumption Diagram

We present a hybrid reasoning system that takes the advantages of both types of reasoners for efficient classification on large and complex ontologies in expressive DLs. Here "complex ontologies" refers to the ontologies which contains a considerable amount of cyclic definitions, which usually causes large models constructed by the tableau procedures. In our approach, for the *main* reasoner we choose one that supports a less expressive language, which we call the base language. From the original ontology $\mathcal{O}$, we first remove the axioms that are beyond the base language, and so construct a weakened ontology $\mathcal{O}_{wk}$. In the second stage, we then inject into $\mathcal{O}_{wk}$ additional axioms to simulate the effects of those removed axioms in a model expansion process, constructing the strengthened ontology $\mathcal{O}_{str}$. These injected axioms are expressed in the base language so they may not perfectly represent the original axioms. We call the stages weakening and strengthening, respectively. After applying these changes to the ontology, we still would like the subsumptions in $\mathcal{O}_{str}$ to contain all the subsumptions in $\mathcal{O}$. In Fig. 1, the results of classification, named the class hierarchy, for ontology $\mathcal{O}$, $\mathcal{O}_{wk}$, and $\mathcal{O}_{str}$ are denoted by H_O, H_{wk} and H_{str} respectively. The subsumptions pairs in H_{str} may be unsound with respect to $\mathcal{O}$. If this occurs, we will need again to verify these suspected pairs by reasoning in the language of the given ontology to remove any unsound subsumptions and other maintenance tasks. We name the reasoner that accepts the full language of $\mathcal{O}$, the *assistant* reasoner; it is potentially slower than the main reasoner.

Our main contributions are as follows:

Hybrid Reasoning using Weakening and Strengthening Approach: We propose a new hybrid reasoning approach by combining the tableau-based and consequence-based reasoning procedures for efficient classification on large and complex ontologies. Concretely, the hybrid reasoning is based on a weakening and strengthening approach and applied to classifying $\mathcal{ALCHOI}$ ontologies, for which we choose $\mathcal{ALCH}$ as the base language to take advantage of the recently developed consequence-based reasoning technique which is able to classify non-Horn ontologies [11].

Implementation and Evaluation: Our system is able to classify ontologies in DL $\mathcal{ALCHOI}$, which is not fully supported by any current consequence-based reasoner. We evaluate our procedure with nine available, practical $\mathcal{ALCHOI}$ ontologies. Except for one ontology we have not gotten the classification result, we are able to achieve soundness with no fewer subsumptions and a better performance than the tableau-based reasoners on large ontologies.

2 Preliminaries and Related Work

The syntax of $\mathcal{ALCHOI}$ uses mutually disjoint sets of *atomic concepts* N_C, *atomic roles* N_R and *individuals* N_I. The set of *roles* is $N_R \cup \{R^- \mid R \in N_R\}$. The set of *concepts* contains A, $\top$, $\bot$, $\neg C$, $C \sqcap D$, $C \sqcup D$, $\exists R.C$, $\forall R.C$, $\{a\}$, for $\top$ the top concept, $\bot$ the

bottom concept, A an atomic concept, C and D concepts, R a role, a an individual. We define $N_C^{\top} = N_C \cup \{\top\}$ and $N_C^{\top,\bot} = N_C^{\top} \cup \{\bot\}$. An ontology $\mathcal{O}$ consists of a set of *general concept inclusions* $C \sqsubseteq D$ and *role inclusions* $R \sqsubseteq S$. A concept equivalence $C = D$ is a shortcut for $C \sqsubseteq D$ and $D \sqsubseteq C$.

An interpretation $\mathcal{I}$ of $\mathcal{O}$ is a pair $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ where $\Delta^{\mathcal{I}}$ is a non-empty set, $\cdot^{\mathcal{I}}$ maps each $A \in N_C$ to a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, each $R \in N_R$ to a relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ and each $a \in N_I$ an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. The interpretation of concepts are defined in [2]. An interpretation $\mathcal{I}$ satisfies axioms $C \sqsubseteq D$ and $R \sqsubseteq S$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ and $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$, respectively. $\mathcal{I}$ is a model of $\mathcal{O}$ if $\mathcal{I}$ satisfies every axiom in $\mathcal{O}$. If every model of $\mathcal{O}$ satisfies an axiom α, we say $\mathcal{O}$ entails α and write $\mathcal{O} \models \alpha$.

An ontology classification task is to compute the class hierarchy $H_{\mathcal{O}}$ containing all the pairs $\langle A, B\rangle$ such that $A, B \in N_C^{\top,\bot}$ and $\mathcal{O} \models A \sqsubseteq B$. We define the role hierarchy H_{op} as the pairs $\langle R, S\rangle$ such that $R, S \in N_R \cup \{R^- \mid R \in N_R\}$ and $\mathcal{O} \models R \sqsubseteq S$.

Our work can be placed in the Theory Approximation setting [10], where a theory Σ is approximated by a lower bound Σ_{lb}, whose models are a subset of the models of Σ, and an upper bound Σ_{ub} whose models are a superset. Our weakening step creates $\mathcal{O}_{wk}$ which is an upper bound Σ_{ub}. Instead of creating a lower bound Σ_{lb}, the target of our strengthening step is to generate an $\mathcal{O}_{str}$ of which some "important" models can be transformed to models of $\mathcal{O}$ so that completeness can be achieved. The details will be explained in Section 4.2. Subsumption results from $\mathcal{O}_{wk}$ are guaranteed to be sound, exactly as queries asked of Σ_{ub} that return "yes" can be taken also as "yes" from Σ. New candidate subsumption results from $\mathcal{O}_{str}$ need to be checked, analogously as queries Σ_{lb} that return "yes" need to be checked.

TrOWL [9] is a soundness-preserving approximate reasoner offering tractable classification for $\mathcal{SROIQ}$ ontologies by an encoding into $\mathcal{EL}^{++}$ with additional data structures. Instead of merely preserving soundness, our algorithm also aims to achieve completeness, although we have not yet proven it. Another difference lies in that the classification procedure of TrOWL is an extension of [1], while our procedure treats both the main and the assistant reasoners as black boxes without changing them.

3 System Overview

The diagram of our system is shown in Fig. 2. The input is an OWL 2 ontology in any syntax supported by the OWL API.[3] The output is the class hierarchy $H_{\mathcal{O}}$ that can be accessed through the OWL API reasoning interfaces. We explain all the components in the following, among which the ones in white boxes are mainly implemented by us:

- The *preprocessor* rewrites some axioms containing constructors that are not supported by the *main reasoner*.
- The *indexer* normalizes the ontology $\mathcal{O}$ and builds an internal representation of it which is suitable for finding axioms and concept expressions. The index speeds up search for strengthening axioms.
- The *axiom injector* calculates the strengthening axioms that approximate the axioms in $\mathcal{O} \setminus \mathcal{O}_{wk}$. The algorithm will be illustrated in Section 4.

[3] Since our algorithm is designed for DL $\mathcal{ALCHOI}$, the unsupported anonymous concepts are replaced with artificial atomic concepts and the unsupported axioms are ignored.

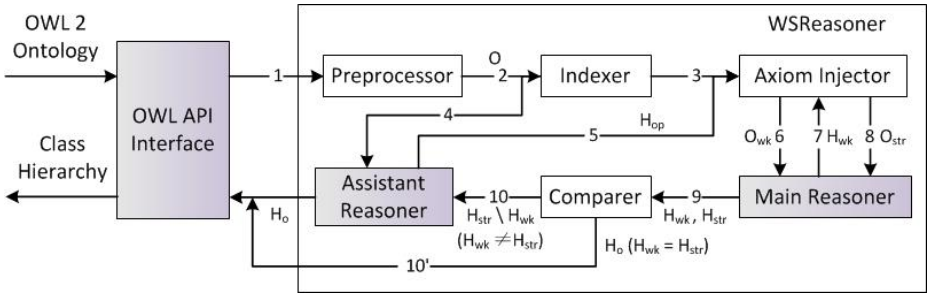

Fig. 2. Key components of WSReasoner

- The *main* and *assistant reasoners* perform the main reasoning tasks. They can be customized by the settings in the configuration instance of each *WSReasoner* object.
- The *comparer* calculates the difference between two concept hierarchies produced by the first and second round of classifications, H_{wk} and H_{str} respectively.

The arrows in the Fig. 2 represent the data flow of the overall reasoning procedure. The numbers on the arrow denote the execution order, and the symbols represent the data. The arrows between the axiom injector and the main reasoner indicates their interactions with each other.

4 The Hybrid Classification Procedure

In this section we give details of the hybrid classification procedure used in WSReasoner. The major phases include preprocessing, normalization and reasoning. Section 4.1 explains preprocessing and normalization. Section 4.2 gives a model-theoretic illustration of the weakening and strengthening approach using an example. And section 4.3 provides the details of the overall procedure and strengthening algorithms.

4.1 Preprocessing and Normalization

In the preprocessing phase, we rewrite the original ontology to make nominals and inverse roles occur only in the axioms of the forms $N_a = \{a\}$ and $R = R'^{-}$, respectively. For nominals, we first rewrite the related OWL 2 DL class expressions and axioms by their equivalent forms containing only singleton nominal concepts, according to Table 1. After that, for each $\{a\}$, we replace all its occurrences by a new concept N_a and add an axiom $N_a = \{a\}$. We call N_a a nominal placeholder for a in the following sections. For inverse roles, we replace each occurrence of R'^{-} in an axiom by a named role R and add an axiom $R = R'^{-}$.

After preprocessing, apart from the axioms of the forms $N_a = \{a\}$ and $R = R'^{-}$, the remaining axioms in the ontology are in DL $\mathcal{ALCH}$. These axioms are normalized using a procedure identical to [11]. The result ontology O contains axioms of the forms $\bigsqcap A_i \sqsubseteq \bigsqcup B_j$, $A \sqsubseteq \exists R.B$, $\exists R.A \sqsubseteq B$, $A \sqsubseteq \forall R.B$, $R \sqsubseteq S$, $N_a = \{a\}$ and $R = R'^{-}$, where A, B are atomic concepts and R, S, R' are atomic roles.

Table 1. Rewriting Nominals in OWL 2

OWL 2 Syntax	Equivalent Forms
Class Expressions	
ObjectHasValue (R a)	$\exists R.\{a\}$
ObjectOneOf ($a_1 \ldots a_n$)	$\{a_1\} \sqcup \ldots \sqcup \{a_n\}$
Axioms	
ClassAssertion (C a)	$\{a\} \sqsubseteq C$
SameIndividual ($a_1 \ldots a_n$)	$\{a_1\} = \ldots = \{a_n\}$
DifferentIndividuals ($a_1 \ldots a_n$)	$\{a_i\} \sqcap \{a_j\} = \bot$
	$1 \leq i < j \leq n$
ObjectPropertyAssertion (R a b)	$\{a\} \sqsubseteq \exists R.\{b\}$
NegativeObjectPropertyAssertion (R a b)	$\{a\} \sqcap \exists R.\{b\} \sqsubseteq \bot$

4.2 Model-Theoretic View of the Strengthening Step

Before going into the details of the reasoning procedure, we give a model-theoretic explanation of the motivation of the strengthening step. Given an ontology $\mathcal{O}$, we want to create its strengthened version $\mathcal{O}_{str}$ which satisfies $H_{\mathcal{O}} \subseteq H_{str}$. To achieve it, we try to ensure that for each $\langle A, B\rangle \notin H_{str}$, there is a certain model $\mathcal{I}$' of $\mathcal{O}_{str}$ for $A \sqcap \neg B$ which can be transformed to a model $\mathcal{I}$ of $\mathcal{O}$ and $(A \sqcap \neg B)^{\mathcal{I}} \neq \emptyset$, so that $\langle A, B\rangle \notin H_{\mathcal{O}}$. Such models $\mathcal{I}$' and $\mathcal{I}$ can be constructed using the hypertableau calculus (abbreviated as HT-calculus) [8]. In the following we first describe strengthening for nominals, followed by strengthening for inverse roles.

4.2.1 Strengthening for Nominals

Example 1. Consider a normalized ontology $\mathcal{O}$ containing the following axioms:

$$A \sqsubseteq \exists R.E \ (1) \quad A \sqsubseteq C \ (2) \quad C \sqsubseteq \forall R.D \ (3) \quad E \sqsubseteq N_a \ (4) \quad N_a = \{a\} \ (5)$$
$$A \sqsubseteq \exists S.N_a \ (6) \quad \exists S.D \sqsubseteq F \ (7) \quad F \sqsubseteq B' \sqcup B \ (8) \quad D \sqsubseteq G \ (9)$$

In Fig. 3 – 5 we give models of $\mathcal{O}$, $\mathcal{O}_{wk}$ and $\mathcal{O}_{str}$ for the concept $A \sqcap \neg B$ constructed by the HT-calculus. In the figures, each node x denotes an individual and its tags represent the concepts that it belongs to, which we call *labels* of x. Each edge $\langle x, y\rangle$ denotes a role relation between two individuals x and y, and its *labels* are the roles that it belongs to. We say that a label B of x is *added by an axiom* α in a normalized ontology if $B(x)$ is added into the model by a derivation corresponding to α in the HT-calculus, e.g. $A \sqsubseteq B$ corresponds to $A(x) \rightarrow B(x)$ and $\exists R.A \sqsubseteq B$ corresponds to $R(x, y) \wedge A(x) \rightarrow B(x)$, etc.

Fig. 3 is a model of $\mathcal{O}_{wk}$, which removes the axiom (5) from $\mathcal{O}$. In the model both of the individuals x_2, x_3 have the label N_a. To build a model of $\mathcal{O}$ based on the HT-calculus, x_2 and x_3 need to be merged into one instance to satisfy the axiom (5). After that, labels F and B' of x_1 will be added by the axioms (7) and (8), yielding the model $\mathcal{I}$ in Fig. 4.

Our strengthening step adds additional axioms $N_a \sqsubseteq E$ and $N_a \sqsubseteq D$ to $\mathcal{O}_{wk}$ to simulate the main effect of the merge operation in the HT-calculus, i.e. making all the instances of N_a have the same labels. With these axioms added, labels D, E and G are added to x_3, and labels F and B' of x_1 can be further introduced by the HT-calculus

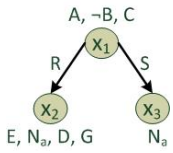

Fig. 3. Model of O_{wk}

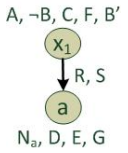

Fig. 4. Model of O

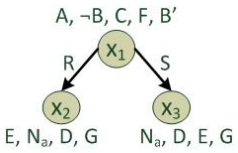

Fig. 5. Model of O_{str}

without the nominal axiom (5). The resulting model $\mathcal{I}$' in Fig. 5 can be transformed to $\mathcal{I}$ by simply merging the instances x_2 and x_3 without extra derivations.

To achieve the above-mentioned effect, we calculate the "important" labels appearing on the instances of N_a in the model of O_{wk}, which we call *major coexisting labels* of N_a. For each instance x, such an important label is the label when x is created by the HT-calculus or a label added by an axiom $A \sqsubseteq \forall R.B$. In other words, these labels are added at initialization time or through a derivation which takes a label on a predecessor of x as a premise, thus they cannot be introduced based on x's own labels. Note that the label G of x_2 in Fig. 3 is not a major coexisting label since it is added by the axiom $D \sqsubseteq G$. For each major coexisting label X of N_a, we choose either $N_a \sqsubseteq X$ or $N_a \sqcap X \sqsubseteq \bot$ as the strengthening axiom, so that X is either added to or prohibited on all the instances of N_a in the model $\mathcal{I}$' of O_{str}. With these axioms added, all the instances of N_a in $\mathcal{I}$' are likely to have identical labels so that $\mathcal{I}$' can be easily transformed to $\mathcal{I}$ to prove $O \not\models A \sqsubseteq B$.

4.2.2 Strengthening for Inverse Roles Regarding inverse roles, the corresponding derivation of an axiom $R = R'^{-}$ in HT-calculus adds $R(x, y)$ if $R'(y, x)$ exists, or vice versa. The new assertion $R(x, y)$ may lead to the following types of further derivations: (1) a label B is added to x by $\exists R.A \sqsubseteq B$; (2) a label B is added to y by axioms $A \sqsubseteq \forall R.B$; (3) labels are added to edges through axioms $R \sqsubseteq S$ and $S = S'^{-}$. To simulate these effects without deriving $R(x, y)$ using $R = R'^{-}$, the following types of axioms are added respectively: (1) $A \sqsubseteq \forall R'.B$; (2) $\exists R'.A \sqsubseteq B$; (3) all the role subsumptions based on the computed role hierarchy H_{op}. Similar axioms need to be added for the assertion $R'(y, x)$. With these axioms added, the model $\mathcal{I}$' of O_{str} can be transformed to a model $\mathcal{I}$ of O by simply satisfying $R = R'^{-}$ without extra derivations. Notice that all the strengthening axioms to handle inverse roles are implied by O, so we have $O \models O_{str}$ and $H_{str} \subseteq H_O$, thus $H_{str} = H_O$ holds and no verifications are needed.

4.3 Classification Procedure

Algorithm 1 gives the overall classification procedure for an $\mathcal{ALCHOI}$ ontology using the weakening and strengthening approach. In the procedure, MR is the main reasoner that provides efficient classification on an $\mathcal{ALCH}$ ontology, while AR is the assistant reasoner, which is slower but capable of classifying the original $\mathcal{ALCHOI}$ ontology O. Function `classify` computes the class hierarchy.

After normalization, the algorithm computes the role hierarchy H_{op}. Line 3 and 4 compute the strengthened ontology O_{istr} for inverse roles, which has the same hierarchy as O. To classify the $\mathcal{ALCHO}$ ontology O_{istr}, we get its weakened and strengthened

versions $\mathcal{O}_{wk}$ and $\mathcal{O}_{str}$ for nominals and classify them with MR, as shown in lines 5 to 8. Computations of $\mathcal{O}_{\mathcal{I}}^{+}$ and $\mathcal{O}_{N}^{+}$ will be explained in Sections 4.3.1 and 4.3.2. Subsumptions in $H_{str} \setminus H_{wk}$ are verified by AR in line 11 to 15. Note that if some $A \sqsubseteq \bot$ is disproved in line 12, $A \sqsubseteq B$ needs to be verified for almost every B in N_C. In this case the workload of verification for AR may exceed that of classifying $\mathcal{O}$, thus we choose to use AR to get $H_{\mathcal{O}}$ directly. Our approach does not add value in this case. Line 14 to 15 verifies each pair in $H_{str} \setminus H_{wk}$ one by one. The verification process can be further improved using a procedure similar to the optimized KP alorithm [3].

Algorithm 1: Classify an $\mathcal{ALCHOI}$ ontology $\mathcal{O}$ using the hybrid approach

Input: An $\mathcal{ALCHOI}$ ontology $\mathcal{O}$
Output: The classification hierarchy $H_{\mathcal{O}}$

1 preprocess and normalize $\mathcal{O}$;
2 H_{op} := AR.classifyObjectProperties($\mathcal{O}$);
3 $\mathcal{O}_{\mathcal{I}}^{+}$:= getStrAxiomsForInverseRoles($\mathcal{O}$, H_{op});
4 $\mathcal{O}_{istr}$:= $\mathcal{O} \cup \mathcal{O}_{\mathcal{I}}^{+}$ with inverse role axioms $R = R'^{-}$ removed;
5 $\mathcal{O}_{wk}$:= $\mathcal{O}_{istr}$ with nominal axioms $N_a = \{a\}$ removed;
6 $\mathcal{O}_{N}^{+}$:= getStrAxiomsForNominals($\mathcal{O}_{istr}$, H_{op});
7 H_{wk} := MR.classify($\mathcal{O}_{wk}$);
8 H_{str} := MR.classify($\mathcal{O}_{wk} \cup \mathcal{O}_{N}^{+}$);
9 remove any $\langle A, B\rangle$ from H_{wk} and H_{str} if $A \notin N_C^{\top,\bot}$ or $B \notin N_C^{\top,\bot}$;
10 $H_{\mathcal{O}}$:= H_{wk};
11 **foreach** $\langle A, \bot\rangle \in H_{str} \setminus H_{wk}$ **do**
12 | **if** AR.isSatisfiable($\mathcal{O}$, A) **then return** AR.classify($\mathcal{O}$);
13 | **else** add $\langle A, \bot\rangle$ into $H_{\mathcal{O}}$;
14 **foreach** $\langle A, B\rangle \in H_{str} \setminus H_{wk}$ **do**
15 | **if** *not* AR.isSatisfiable($\mathcal{O}$, $A \sqcap \neg B$) **then** add $\langle A, B\rangle$ into $H_{\mathcal{O}}$;
16 **return** $H_{\mathcal{O}}$

4.3.1 Strengthening for Inverse Roles Based on the discussions in Section 4.2, we calculate the strengthening axioms $\mathcal{O}_{\mathcal{I}}^{+}$ for inverse roles according to the following steps:

1. For each $\langle R', S^{-}\rangle \in H_{op}$ where S^{-} does not have an equivalent named role, introduce a new named role S' for S^{-} and update H_{op}.
2. Initialize $\mathcal{O}_{\mathcal{I}}^{+}$ with all the subsumptions between named roles in H_{op}
3. for each $\langle R, R'\rangle$ such that $R = R'^{-}$ is implied by H_{op}, if either of the following two equivalent forms is used, add the other to $\mathcal{O}_{\mathcal{I}}^{+}$:

$$A \sqsubseteq \forall R.B \Leftrightarrow \exists R'.A \sqsubseteq B$$

Here $\langle R, R'\rangle$ and $\langle R', R\rangle$ are treated as different pairs.

4.3.2 Strengthening for Nominals This section explains the calculation of strengthening axioms $\mathcal{O}_{N}^{+}$ for nominals. According to Section 4.2, for each nominal placeholder

N_a, we need to compute its major coexisting label set LS_{N_a}, and choose to add $N_a \sqsubseteq X$ or $N_a \sqcap X \sqsubseteq \bot$ into $\mathcal{O}_N^+$ for each $X \in LS_{N_a}$.

Algorithm 2: Calculate the potential major coexisting label set of N_a in $\mathcal{O}$

Input: Normalized $\mathcal{ALCHOI}$ ontology $\mathcal{O}$ and a concept $N_a \in N_C$
Output: Major coexisting label set LS_{N_a}

1 Initialize a queue Q with label N_a;
2 $Core_{N_a} := \emptyset$; visited $:= \emptyset$;
3 **repeat**
4 poll a label X from Q;
5 **if** *X is not introduced by some* $\bigsqcap A_i \sqsubseteq M \bigsqcup X$ *and* $X \notin$ visited **then**
6 add X to *proc*;
7 **foreach** $\bigsqcap A_i \sqsubseteq M \bigsqcup X \in \mathcal{O}$ **do** add each A_i into Q;
8 **foreach** $\exists S.Y \sqsubseteq X \in \mathcal{O}$ *and* $\langle R, S \rangle \in H_{op}$ *and* $B \sqsubseteq \exists R.Z \in \mathcal{O}$ **do**
9 add B into Q;
10 **foreach** $Y \sqsubseteq \forall S.X \in \mathcal{O}$ *and* $\langle R, S \rangle \in H_{op}$ *and* $B \sqsubseteq \exists R.Z \in \mathcal{O}$ **do**
11 add Z to $Core_{N_a}$;
12 **if** $X \in N_C^\top$ *or some* $B \sqsubseteq \exists R.X \in \mathcal{O}$ **then** add X to $Core_{N_a}$;
13 **until** *Q is empty*;
14 $LS_{N_a} := Core_{N_a}$;
15 **foreach** $X \in Core_{N_a}$ **do**
16 **foreach** $B \sqsubseteq \exists R.X \in \mathcal{O}$ *and* $\langle R, S \rangle \in H_{op}$ *and* $Y \sqsubseteq \forall S.Z \in \mathcal{O}$ **do**
17 add Z to LS_{N_a};
18 **return** LS_{N_a}

Algorithm 2 illustrates the computation of LS_{N_a} in Example 1. From line 3 to 13 we search for the potential core label set $Core_{N_a}$ of N_a, i.e., the concepts that may label an individual of N_a when it is created by the HT-calculus. $Core_{N_a}$ is a subset of LS_{N_a}. We search in the converse direction of the model construction process for the labels X that may cause the appearance of label N_a, which we denote by $X \mapsto N_a$. and put the potential core labels into $Core_{N_a}$. There are three cases that X is added to an individual x according to the calculus:

Case 1: (Line 7) If X is added to x by the axiom $\bigsqcap A_i \sqsubseteq M \sqcup X$, then for every conjunct condition A_i we have $A_i \mapsto X$.

Case 2: (Line 8-9) If X is added to x to by the axiom $\exists S.Y \sqsubseteq X$, then x has an S-successor, which must be introduced by some $B \sqsubseteq R.Z$ provided that $\langle R, S \rangle \in H_{op}$. For every such B there is a potential that $B \mapsto X$.

Case 3: (Line 10) X is added to x by the axiom $Y \sqsubseteq \forall S.X$, then it must have an S-predecessor in the model. Thus when x is created, the incoming role R satisfies $\langle R, S \rangle \in H_{op}$, and for all $B \sqsubseteq \exists R.Z$, Z is a potential core label of x.

Line 12 checks whether X itself can be a core label. A core label can only be introduced through: (1) the initialization step, for which X must be atomic in $\mathcal{O}$; (2) an individual-adding derivation, for which there must be some $B \sqsubseteq \exists R.X \in \mathcal{O}$.

On line 15 to 17 we follow the model construction process to find other major coexisting labels. Similarly to case 3 above, if x has a core label X added by the axiom $B \sqsubseteq \exists R.X$ and $\langle R, S \rangle \in H_{op}$, then Z may be added to X by the axiom $Y \sqsubseteq \forall S.Z$.

The test on line 5 prunes a search branch X in either of two cases: (1) X has been visited. (2) An axiom $\bigsqcap A_i \sqsubseteq M \sqcup X$ has been used on the search path from N_a to X. In case (2), when the model expands, the axiom $\bigsqcap A_i \sqsubseteq M \sqcup X$ has been satisfied and no new labels that potentially introduces N_a will be added.

We show the calculation of the strengthening axioms for N_a in Example 1. Q is initialized with N_a. E is added to Q according to Case 1 based on the axiom $E \sqsubseteq N_a$. When E is processed, it is added to $Core_{N_a}$ in line 12 and also to LS_{N_a} in line 14. D is added to LS_{N_a} in the next loop from line 15 to 17, based on the axioms $A \sqsubseteq \exists R.E$ and $C \sqsubseteq \forall R.D$. Finally we choose to add $N_a \sqsubseteq D$ and $N_a \sqsubseteq E$ based on some heuristic rules.

Termination of the outer loop from line 3 to 13 is ensured by keeping a visited set so that any label will only be processed at most once in the loop. Let n_c and n_{ax} be the number of concepts and axioms in $\mathcal{O}$. One can see that the inner loop on line 7 runs at most n_{ax} times, while the number of runs of the next two inner loops is bounded by the number n_{ax}^2 of pairs of axioms. So the worst-case complexity of the outer loop is $O(n_c \cdot n_{ax}^2)$. The case is similar for the loop from 15 to 17. This procedure needs to be invoked for each concept N_a, the number of invocations is less than n_c. Since $n_c \leq n_{ax}$ in the normalized ontology, the worst-case number of executions is polynomial in n_{ax}.

5 Evaluation

We have implemented our proposed approach in a prototype reasoner WSReasoner. We use the consequence-based reasoner ConDOR r.12 as the main reasoner and the hyper-tableau-based reasoner HermiT 1.3.6[4] as the assistant reasoner. ConDOR supports DL $\mathcal{SH}$ ($\mathcal{ALCH}$ + transitivity axioms), and HermiT supports DL $\mathcal{SROIQ}(\mathcal{D})$, which is more expressive than the $\mathcal{ALCHOI}$. We compared the performance of WSReasoner with the latest versions of mainstream reasoners, including tableau-based reasoners HermiT 1.3.6, Fact++ 1.5.3 and Pellet 2.3.0, as well as a consequence-based reasoner TrOWL 0.8.2. All the experiments were run on a laptop with an Intel Core i7-2670QM 2.20GHz quad core CPU and 16GB RAM running Java 1.6 under Windows 7. We set the Java heap space to 12GB. We did not set the time limit.

We tried all the commonly used, widely available large and complex ontologies that we have access to. Since none of these expressive ontologies are modeled in $\mathcal{ALCHOI}$, we had to make some adjustments. Galen[5] and FMA-constitutionalPartForNS (FMA-cPFNS) are currently available large and complex ontologies. We use three different version of Galen, which are Full-Galen, Galen-Heart, and Galen-EL.[6] We modify them by introducing nominals. In Galen, the concepts starting with a lower case letter and subsumed by *SymbolicValueType* could be nominals which are modeled as concepts. For Galen-Heart and Galen-EL, we used published methods [7] to produce two versions for each of them, which are Galen-Heart-YN1, Galen-Heart-YN2, Galen-EL-YN1, and

[4] HermiT 1.3.6 build 1054, 04/18/2012 release

[5] http://www.co-ode.org/galen/

[6] http://code.google.com/p/condor-reasoner/downloads/list/

Galen-EL-YN2. In addition, we produced Galen-EL-LN1 by introducing norminals only for leaf N-concepts of Galen-EL; and also add disjunction into Full-Galen and produced Galen-Full-UnionN2.

We also used two smaller complex ontologies, Wine and DOLCE. All our ontologies were reduced to $\mathcal{ALCHOI}$ and can be downloaded from our website.[7]

Table 2. Comparison of classification performance

T: Time(seconds); MS: (# of subsumption pairs missing) / (# of total subsumption pairs)

Ontology	**Criteria**	**(Hyper) tableau**			**Consequence-based**	**WSReasoner**	
		HermiT	**Pellet**	**FaCT++**	**TrOWL**	**com-role**	**fast-role**
Wine	T	29.11	377.88	7.33	0.81	7.34	1.22
	MS	0/968	0/968	0/968	1/968	0/968	0/968
DOLCE	T	5.64	8.40	0.92	0.55	8.84	2.09
	MS	0/2595	0/2595	0/2595	390/2595	0/2595	0/2595
Galen-Heart-YN1	T	115.10	-	-	-	7.59	6.94
	MS	0/45,513	-	-	-	0/45,513	0/45,513
Galen-Heart-YN2	T	63.52	-	-	-	9.50	7.19
	MS	0/45,914	-	-	-	0/45,914	0/45,914
Galen-EL-YN1	T	197,090	-	-	-	345,600+	345,600+
	MS	0/431,990	-	-	-	/	/
Galen-EL-YN2	T	289,637	-	-	-	38,350	38,272
	MS	0/457,779	-	-	-	0/457,779	0/457,779
Galen-EL-LN1	T	188,018	-	-	-	771	755
	MS	0/431,990	-	-	-	0/431,990	0/431,990
Galen-Full-UnionN2	T	604,800+	-	-	-	1625	1478
	MS	/	-	-	-	x/(431,255+x)	x/(431,255+x)
FMA-cPFNS	T	667,430	-	-	429.65	21,362	45
	MS	0/481,967	-	-	70/481,967	0/481,967	0/481,967

Note: "-" entry means that the reasoner was unable to classify the ontology due to some problems. "/" entry means the number is not available.

The results of our experiment are shown in Table 2. Since the time limit is not set, some tasks may take several days or more to finish. The '+' sign indicates the tasks are not finished within the time shown before '+'. We also report the number of missed subsumptions, since some of the reasoners are not complete, such as TrOWL and possibly ours. # represents number in Table 2 and 3. The total number includes all pairs of subsumptions between any $A, B \in N_C^{\top,\bot}$ except for $\bot \sqsubseteq A$ and $A \sqsubseteq \top$. The complete result is obtained from HermiT. For Galen-Full-UnionN2, HermiT does not get the results, while WSReasoner gets 431,255 subsumptions, but we do not know the number of missed pairs, so we denote it by x.

Since computing complete role hierarchy H_{op} takes a considerable amount of time for our assistant reasoner HermiT, we implement two versions of WSReasoner. The version 'fast-role' simply computes the reflexive-transitive relations between roles,[8] while the version 'com-role' request HermiT for a complete H_{op} on O.

[7] http://isew.cs.unb.ca/wsreasoner/resources/ontologies/

[8] to simplify implementation, we extract all object property axioms and request HermiT for H_{op}

As we can see in Table 2, WSReasoner is able to classify eight of nine ontologies, and get no fewer subsumptions than any other reasoners. On various FMA-cPFNS and versions of Galen ontologies, WSReasoner outperforms all the other reasoners considerably except for Galen-EL-YN1, on which the verification stage has more than 320,000 pairs to verify, however, it takes only several minutes to determine that WSReasoner is likely to be slower than HermiT. For the two smaller ontologies Wine and DOLCE, the fast-role approach still outperforms HermiT and Pellet and even com-role outperforms the two reasoners on Wine.

Table 3. Statistics of WSReasoner

Added-axioms: # of axioms in $\mathcal{O}_N^+$ Add-pairs: # of pairs in $H_{str} \setminus H_{wk}$
True-pairs: # of pairs verified to be correct Verify-time: verification time
fast-role-T: The role classification time using the fast-role approach (Seconds)
com-role-T: The role classification time using the comp-role approach (Seconds)

Ontology	Wine	DOLCE	Galen-Heart-YN1	Galen-Heart-YN2	Galen-EL-YN1	Galen-EL-YN2	Galen-EL-LN1	Galen-Full-UnionN2	FMA-cPFNS
Added-Axioms	36	95	2	50	79,594	981	12	1100	0
Add-Pairs	0	0	2	403	323,294	28,287	45	259	0
True-Pairs	0	0	2	403	-	25,789	0	0	0
fast-role-T	0.16	0.77	0.639	0.56	1.60	1.25	1.63	2.82	0.39
com-role-T	6.50	7.37	2.09	2.53	22.99	19.57	19.37	15.30	21322.54
Verify-Time	0	0	3.39	3.84	-	38,252	649.24	793.26	0

Table 3 shows some statistics of our reasoner on different phases. FMA-cPFNS only needs one round of classification. Wine and DOLCE need two rounds of classification but no verifications are needed, which indicates the nominals does not bring any new subsumptions. The number of strengthening axioms added for these ontologies varies a lot. The verification time becomes larger as the size of $H_{str} \setminus H_{wk}$ increases. Comparing the role classification time, fast-role is considerably faster than com-role, especially for the ontology FMA-cPFNS on which the com-role approach takes 5 hours to finish. In summary, the main factors affecting the reasoning time are: (1) The approach to choose for role hierarchy calculation. (2) the number of pairs in $H_{str} \setminus H_{wk}$.

To evaluate the performance on other existing ontologies, we also ran WSReasoner on a test suite provided by ORE 2012.[9] We used the RDF version of the ontologies in the OWL DL and OWL EL datasets, which contains 107 and 8 real-world ontologies respectively. We set the maximum Java heap space to 1GB and add an additional JVM setting `-DentityExpansionLimit=480000` to ensure successful loading of all the ontologies using OWL API. The time limit is set to 1 hour.

The results are shown in Table 4. The columns "OWL DL" and "OWL EL" refer to the two datasets, while "DL-ALCHOI" and "EL-ALCHOI" refer to the $\mathcal{ALCHOI}$ ontologies in these datasets, respectively. As seen in the table, all the $\mathcal{ALCHOI}$ ontologies have been correctly classified. Complete hierarchies of 5 ontologies in the OWL

[9] http://www.cs.ox.ac.uk/isg/conferences/ORE2012/datasets/classification.zip

DL dataset are not obtained because their languages are beyond $\mathcal{ALCHOI}$. The largest ontology `gazetteer` in the OWL DL dataset (containing 150979 classes) causes a out-of-memory exception.

Table 4. Evaluation results on the ORE test suite

ALT: Average Loading Time ART: Average Reasoning Time
TOTAL: # of ontologies in the dataset
CORRECT: # of correctly classified ontologies
INCORRECT: # of incorrectly classified ontologies
NO-REF: # of ontologies with no reference results
EXCEPTION: # of ontologies that cause an exception
TIMEOUT: # of ontologies that cause a timeout

Outcomes	**DL-ALCHOI**	**OWL DL**	**EL-ALCHOI**	**OWL EL**
ALT (ms)	105	323	320	454
ART (ms)	632	1263	277	503
TOTAL	34	107	6	8
CORRECT	18	47	5	6
INCORRECT	0	5	0	0
NO-REF	16	54	1	2
EXCEPTION	0	1	0	0
TIMEOUT	0	0	0	0

6 Conclusions and Future Work

We present an approach combining two reasoners based on ontology weakening and strengthening to classify large and complex ontologies, for which tableau-based reasoners may take a long time or use much memory while consequence-based reasoners cannot support all the constructors in the target language. We use a consequence-based reasoner supporting a DL less expressive than the target language as the main reasoner to do the majority (sometimes all) of the work, and a more expressive but slower tableau-based reasoner to assist it in verifying the results. In the experiment dataset shown in Table 2, WSReasoner's results show better efficiency than the tableau-based reasoners in most large and complex ontologies, and no fewer subsumptions than other reasoners except for one ontology for which the result is not obtained by WSReasoner.

The weaknesses of this WSreasoner for $\mathcal{ALCHOI}$ are: (1) the complete role hierarchy classification may take a lot of time; (2) if the number of pairs verified is large, the procedure still takes a lot of time in the verification stage. The advantages of the WS approach are: (1) it may achieve better classification efficiency than tableau-based reasoners; (2) it extends the capability of consequence-based reasoners; (3) the reasoners underneath can be customized to classify different ontologies.

In the future, we will try to prove the theoretical completeness of this weakening and strengthening approach for $\mathcal{ALCHOI}$ ontology classification while optimizing the

algorithm. We will further try to apply the weakening and strengthening approach based on different reasoners and address more constructors for more expressive languages.

References

1. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: IJCAI 2005, Proceedings of the 19th International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30-August 5, 2005. pp. 364–369 (2005)
2. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge Univ Pr (2010)
3. Glimm, B., Horrocks, I., Motik, B., Stoilos, G.: Optimising ontology classification. In: Patel-Schneider, P., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J., Horrocks, I., Glimm, B. (eds.) The Semantic Web - ISWC 2010, Lecture Notes in Computer Science, vol. 6496, pp. 225–240. Springer Berlin / Heidelberg (2010)
4. Horrocks, I., Sattler, U.: A tableau decision procedure for $\mathcal{SHOIQ}$. Journal of Automated Reasoning 39, 249–276 (October 2007)
5. Kazakov, Y.: Consequence-driven reasoning for horn $\mathcal{SHIQ}$ ontologies. In: IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009. pp. 2040–2045 (2009)
6. Kazakov, Y., Krötzsch, M., Simančík, F.: Concurrent classification of $\mathcal{EL}$ ontologies. In: ISWC 2011, 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011. Lecture Notes in Computer Science, vol. 7031, pp. 305–320. Springer (2011)
7. Kazakov, Y., Krötzsch, M., Simančík, F.: Practical reasoning with nominals in the $\mathcal{EL}$ family of description logics. In: Proceedings of the 13th International Conference on Principles of Knowledge Representation and Reasoning (KR'12) (2012)
8. Motik, B., Shearer, R., Horrocks, I.: Hypertableau reasoning for description logics. Journal of Artificial Intelligence Research 36, 165–228 (September 2009)
9. Ren, Y., Pan, J.Z., Zhao, Y.: Soundness preserving approximation for TBox reasoning. In: AAAI 2010, Proceedings of the 24th AAAI Conference on Artificial Intelligence, Atlanta, Georgia, USA, July 11-15, 2010. AAAI Press 2010 (2010)
10. Selman, B., Kautz, H.: Knowledge compilation and theory approximation. Journal of the ACM (JACM) 43(2), 193–224 (1996)
11. Simančík, F., Kazakov, Y., Horrocks, I.: Consequence-based reasoning beyond horn ontologies. In: IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, 2011. pp. 1093–1098 (July 2011)
12. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A Practical OWL-DL Reasoner. Web Semantics: Science, Services and Agents on the World Wide Web 5(2), 51–53 (2007)
13. Tsarkov, D., Horrocks, I.: FaCT++ Description Logic Reasoner: System Description. Automated Reasoning pp. 292–297 (2006)