Project Report
On

Context Labeling and Context-based Reasoning on
semantic web ontologies

Submitted to

Dr. Rafael Peñaloza Nyssen
Theoretical Computer Science Group, TU Dresden, Germany

By

T. Aparna Saisree
March 2014
Introduction

Description Logics are a family of logical formalisms, which can be used to represent the conceptual knowledge of an application domain in a structured and formally well-understood way. From the DL point of view, (1) an ontology is a finite set of axioms, which formalize our knowledge about the relevant concepts of the application domain. As ontologies grow in size, a common defect found in OWL Ontologies is unsatisfiable concepts, i.e. concepts which cannot have any individuals. Unsatisfiable concepts or error-prone consequences are usually a fundamental modeling error, and are also quite easy for a reasoner to detect and for a tool to display. However, determining why a concept in an ontology is unsatisfiable can be a considerable challenge.

In this project, we have developed a software tool to show how an ontology can be labeled with contexts so that unwanted consequences are not entailed from any context of the ontology. We have implemented the black-box approach based on Reiter’s Hitting set algorithm to find the minimal set of repairs for the erroneous consequences.

The application of this tool is error tolerant reasoning can be done over context labeled ontologies, i.e. we can compute from boundary of a consequence from the ontology.

Following algorithms are implemented in this tool to create labeled ontology and to do context-based reasoning over it.

Basic Definitions

Definition 1.1

Let $T$ be a TBox and $C, D$ two concepts. A minimal axiom set ($\text{MinA}$) for $C \sqsubseteq_T D$ is a TBox $\mu \subseteq T$ such that:

$C \sqsubseteq_{\mu} D$ and

For every $S \not\subseteq \mu$, it holds that $C \not\sqsubseteq_S D$

Definition 1.2

A diagnosis for $O, c$ is a sub-ontology $S \subseteq O$ such that $O \setminus S \neq c$ for all $S' \subseteq S$.

The dual notion of a MinA is that of a diagnosis.

Definition 1.3

A repair for $O, c$ is sub-ontology $R \subseteq O$ such that if $S$ is a diagnosis then $O \setminus S$ is the repair.
Definition 1.4

Let $O$ be an ontology, $\text{lab}$ a labeling function and $c$ a consequence. An element $v \in L$ is called a **boundary** for $O$, $c$, $\text{lab}$ if for every join prime element relative to $L_{\text{lab}}$ it holds that $\ell \leq v$ iff $O \models \ell \iff c$.

Implemented Algorithms

**Computing minimal set of repairs based on Reiter’s Hitting set algorithm:**

Based on the idea of axiom-pinpointing and using black-box reasoning, we get one $\text{minA}$ of the consequence at a time. From the $\text{minA}$ we remove one axiom at a time from the ontology and compute if the consequence is still entailed from the ontology, and then compute next $\text{minA}$. This process continues till sufficient axioms are removed so that the consequence is not entailed from the ontology. Then we get one diagnosis for the consequence. Repair is computed as the complement of the diagnosis. Then we do backtracking, put back one axiom at time in the ontology and remove next axiom from the current $\text{minA}$ and check entailment of the consequence. In this way all the repairs are computed for the consequence.

Once the repairs are computed then we keep only the minimal set of repairs and also remove duplicates.

To compute contexts we check to which set of repairs, each axiom belongs to. The context label assigned to axiom is the set of repairs it belongs to. If the axiom is not part of any repair then that axiom will not have context label. So the error-prone axiom will not be part of any repair and it will not have a context label. So it does not follow from any context of the ontology.

If the input consequence is not entailed from the ontology then all the axioms of the ontology belong to one repair and all axioms in the ontology will have one context label.

**Margin – Based Boundary Algorithm**

Margin-based Boundary computation is one of the context based reasoning tasks. It is computed based on full Axiom Pinpointing and Black-box approach. The boundary of a consequence is the minimal contexts from which the consequence entails from the ontology.

This computation is implemented as follows; first we connect to a reasoner and get all $\text{MinAs}$ for the consequence. Depending on the context labels of the $\text{MinAs}$, $\text{MinLabels}$ are computed. Then margin for the consequence is computed from each set of $\text{MinLabels}$. Margin is the supremum of the Set of $\text{MinLabels}$. And then boundary for the consequence is computed as the infimum of the margins obtained.
This is purely black-box approach, the efficiency of the algorithm depends on the efficiency of the reasoner plugged-in, with the efficient reasoner which can compute all the MinAs fast, this algorithm is efficient. The efficiency also depends on the number of MinAs for the consequence.

**Hitting Set Tree Algorithm**

Hitting Set Tree (HST) Boundary computation is another kind of context based reasoning tasks. The HST-based method for axiom pinpointing computes one MinA at a time while building a tree that expresses the distinct possibilities to be explored in the search of further MinAs. It first computes an arbitrary MinA $S_0$ for $O$, which is used to label the root of the tree. Then, for every axiom $a$ in $S_0$, a successor node is created. If $O \setminus \{a\}$ does not entail the consequence, then this node is a dead end. Otherwise, $O \setminus \{a\}$ still entails the consequence. In this case, a MinA $S_1$ for $O \setminus \{a\}$ is computed and used to label the node. The MinA $S_1$ for $O \setminus \{a\}$ obtained this way is also a MinA of $O$, and it is guaranteed to be distinct from $S_0$ since $a \notin S_1$. Then, for each axiom $a'$ in $S_1$, a new successor is created, and treated in the same way as the successors of the root node, i.e., it is checked whether $O \setminus \{a, a'\}$ still has the consequence, etc. This process obviously terminates since $O$ is a finite set of axioms, and the end result is a tree, where each node that is not a dead end is labeled with a MinA, and every existing MinA appears as the label of at least one node of the tree.

An important ingredient of the HST algorithm is a procedure that computes a single MinA from an ontology. Such a procedure can, e.g., be obtained by going through the axioms of the ontology in an arbitrary order, and removing redundant axioms, i.e., ones such that the ontology obtained by removing this axiom from the current subontology still entails the consequence.

**Implementation**

We have implemented the above discussed algorithms, i.e., Computing repairs using axiom pinpointing algorithm for labeling the ontology. Margin based boundary algorithm and Hitting Set Tree based Boundary algorithm for computing boundary for a consequence over labeled ontology. We have developed a new PROTÉGÉ plug-in called COBRA with this implementation. COBRA processes concept descriptions and ontologies in OWL API format.

**Labeling ontology**

We now focus on the issue of creating context labels in the ontology. Context is added as annotation property to the ontology and every axiom of the ontology that belongs to some contexts will be assigned with a context label. Axioms that do not belong to any context will remain unchanged. Implementation details are discussed below.

Algorithm 1 to algorithm 4 describes how repairs are computed for a given (unwanted) consequence of the ontology based on Reiter’s Hitting set algorithm and how contexts are assigned to axioms of the ontology.
Procedure \textit{labelOntology} \((O, c)\) takes unlabeled ontology \(O\), consequence \(c\), as input and output is the context labeled ontology \(O\). The procedure proceeds as follows, the global variable \(R\) refers to the set of repairs. Initially \(R\) is empty set, if ontology do not entail the consequence then whole ontology \(O\) is considered as repair for \(c\) and whole ontology is assigned with one context. If consequence follows from ontology then it calls \textit{computeRepairs} procedure to compute repairs. After computing repairs contexts are computed accordingly.

\begin{algorithm}
\caption{LabelOntology}
\textbf{Procedure} labelOntology \((O, c)\)
\begin{algorithmic}
\Input \(O\): ontology; \(c\): consequence;
\Output \(O\): labeled ontology
\begin{enumerate}
\item Global \(R := \emptyset\) \((R\): set of repairs)\)
\item if \(O \not\models c\) then
\item \(R := O;\)
\item Call computeContexts\((R, O)\);
\item else
\item Call computeRepairs\((O,c)\); \(\text{(Call computeRepairs} (O,c))\)
\item \(R := \{r_1,\ldots,r_n\}\)
\item \(O := \text{computeContexts} (R, O)\); \(\text{(Call computeContexts for the repairs obtained)}\)
\item end if
\item return \(O;\)
\end{enumerate}
\end{algorithmic}
\end{algorithm}

Computing repairs based on Reiter’s Hitting set algorithm

Procedure \textit{computeRepairs} \((O, c)\) takes unlabeled ontology \(O\), consequence \(c\) as input and set of repairs \(R\) as output. The procedure computes one \(\text{min} A\) of \(c\) at a time and removes one axiom \(a_i\) at a time from computed \(\text{min} A\) and checks if there exists another \(\text{min} A\). While \(c\) is entailed after removing \(a_1,\ldots,a_i\) from each \(\text{min} A\) then add those axioms to diagnosis set, \(d\). If \(c\) is no more entailed after removing \(a_i\) from \(O\) then \(a_i\) is added back to \(O\), at this stage a repair \(r_i\) is computed as \(O \setminus d\) and it is added to \(R\), then backtrack by removing next axiom from previous \(\text{min} A\). This procedure continues until all axioms of all the computed \(\text{min} A\)s are checked. \(\text{min} A\)\_set stores all the \(\text{min} A\)s and process them. And set of repairs are computed.

\begin{algorithm}
\caption{Compute repairs}
\textbf{Procedure} computeRepairs \((O, c)\)
\begin{algorithmic}
\Input \(O\): Unlabeled ontology, \(c\): consequence
\Output \(R:= \{r_1,\ldots,r_n\} \ (1 \leq i < n)\) \(\text{(R: set of repairs of the ontology for the consequence)}\)
\begin{enumerate}
\item Global \(d:=0, R:= \{r_1,\ldots,r_n\} \ (1 \leq i < n), \text{minA} := \emptyset, \text{minA}\_\text{set} = \emptyset; \text{\(d\): diagnosis; \(\text{minA}\)\_set set of \text{minA}s)}\)
\item \(O_{\text{temp}} := \emptyset; \text{(\(O_{\text{temp}}\): temporary ontology to keep track of changes done in this algorithm)}\)
\item \(O_{\text{temp}} := O;\)
\end{enumerate}
\end{algorithmic}
\end{algorithm}
4. \( \text{minA} := \{a_1, \ldots, a_n\} \); (Get one \( \text{minA} \) of the consequence from \( O \); \( a_1, \ldots, a_i \): set of axioms that form the \( \text{minA} \))

5. \( \text{minAs} := \text{minAs} \cup \text{minA} \);

6. while \( (\text{minA} \neq \emptyset) \) do

7. for each \( a_i \in \text{minA} \)

8. \( O_{\text{temp}} := O_{\text{temp}} / a_i \); (from \( \text{minA} \) remove one axiom at a time from the ontology)

9. if \( O_{\text{temp}} \models c \) then And check if consequence is still entailed from the ontology

10. \( d := d \cup a_i \);

11. Call computeRepairs\( (O, c) \)

12. else

13. \( R := R \cup (O / d) \); (Compute repair)

14. \( O_{\text{temp}} := O_{\text{temp}} \cup a_i \);

15. \( d := d \setminus a_i \);

16. \( \text{minAs} := (\text{minAs} \setminus \text{minA}) \cup (\text{minA} \setminus a_i) \);

17. if \( d = \emptyset \) and \( \text{minAs} = \emptyset \) then

18. return \( R \)

19. else

20. Continue

21. return \( R \);

---

### Compute contexts from repairs

The procedure \( \text{computeContexts} (R, O) \) computes contexts labels for each axiom of the ontology. It takes set of repairs \( R \), as input and checks to which repairs each axiom belongs to. If axiom \( a_i \in \{r_1, \ldots, r_i\} \) then \( a_i \) will get context label as \( \{1, \ldots, i\} \) (\( i \in \mathbb{N} \)). This method also removes duplicates and subsets from set of repairs. If \( R \) is whole ontology \( O \) then all axioms of the ontology will get context label as \( 1 \). After context labels are computed then context label is added to \( a_i \) and \( a_{\text{inew}} \) i.e., axiom with context label is added back to the ontology.

**labelsToAxioms**\( (a, \text{labels}) \), this set creates mapping between every axiom of the ontology and contexts labels assigned to it.

**Algorithm 3 compute contexts**

<table>
<thead>
<tr>
<th>Procedure</th>
<th>computeContexts( (R, O) )</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong></td>
<td>( R := {r_1, \ldots, r_n} ); ( R ): Set of repairs</td>
</tr>
<tr>
<td><strong>Output:</strong></td>
<td>( O ): labeled ontology</td>
</tr>
</tbody>
</table>

1. Global \( \text{labelsToAxioms} (a, \text{labels}) := \emptyset \); (this set creates mapping between axioms and context labels assigned to them, \( a \): axiom, \( \text{labels} \): corresponding labels assigned to \( a \))

2. Global \( \text{axioms} := \{a_1, \ldots, a_n\}, a_{\text{inew}} := \emptyset \); (axioms: set of all axioms of the ontology, \( a_{\text{inew}} \): new context annotated axiom to be added to ontology)

3. if \( R = O \) then

4. for each \( a_i \in \text{axioms} \) (\( 1 \leq i \leq n \)) do

5. \( \text{labels} (a_i) := 1 \);

6. end for;

7. else
8. for each $r_i, r_j \in R, (1 \leq i < j \leq n)$ do
9.  if $r_i == r_j$ then
10.  $R := R \setminus r_i$; (remove duplicates)
11.  else if($r_i \subseteq r_j$) then
12.  $R = R \setminus r_i$; (keep only maximal set of repairs)
13.  end if;
14. labelsToAxioms($a, labels$) = createLabelsToAxioms($R$);
15. end if;
16. for each $a_i \in axioms (1 \leq i \leq n)$ do
17.  $O = O \setminus a_i$; (remove original axiom from ontology)
18.  $a_{\text{new}} = a_i + labels (a_i)$; (add context annotation labels to axiom)
19.  $O = O \cup a_{\text{new}}$;
20. end for;
21. return $O$;

Procedure computeLabelsToAxioms($R$) takes set of repairs as input and computes context label to each axiom of Ontology. Creates mapping between axioms and labels called labelsToAxioms($a, labels$).

Algorithm4 Create labels to axioms

Procedure computeLabelsToAxioms($R$)
Input: $R := \{r_1,\ldots,r_n\};$ (R: Set of repairs)
Output: labelsToAxioms (axioms, labels); (set of axioms and context label assigned to each axiom)

1. Global label =1, axioms:= $\emptyset$;  (axioms: this set contains axioms that belongs to the repair being processed);
2. for each $r_i \in R, (1 \leq i \leq n)$ do
3.  axioms := $\{a_1,\ldots,a_m\} \in r_i$;
4.  for each $a_j \in axioms, (1 \leq j \leq m)$ do
5.  if($a_j \notin labelsToAxioms$) then
6.  labelsToAxioms:= labelsToAxioms $\cup$ labelsToAxioms ($a_j, label$);  (Add axiom and label to the set)
7.  label:= label +1;  (increment the label)
8.  else if($a_j \in labelsToAxioms$) then
9.  Local labelSet := $\emptyset$;  (labelSet: this set contains labels already assigned to the axiom under processing)
10.  labelSet:= labelsToAxioms.get($a_j$);  (this method gets labels of axiom $a_j$)
11.  labelSet = labelSet + label;
12.  labelsToAxioms:= labelsToAxioms $\cup$ labelsToAxioms($a_j, labelSet$);  (Add new set of labels to axiom)
13.  label = label +1;
14. end if
15. end for
16. return labelsToAxioms(axioms, labels);
Context –based Reasoning

After creating context labeled ontology, next task is to do context-based reasoning over it. Here we have implemented two algorithms for reasoning, i.e. to compute boundary of a consequence. Reasoning algorithms implemented are Margin based boundary and Hitting Set Tree (HST) boundary.

Compute Margin-based Boundary

Procedure `calculateMarginBasedBoundary(O, c)` takes Ontology O, consequence c as input. Boundary b of c as output. This procedure gets all minAs for c at once from the reasoner and stores them in `minAs_set := {minA1,......,minAn}`. for each `minAi := {a1,...,am}` (minA is set of axioms with context label) of `minAs_set` the procedure computes minLab of axioms of `minAi`. After minLab of all minAs is computed, infimum of set of supremum is computed as boundary of the c.

Algorithm5 Margin-based boundary

Procedure `calculateMarginBasedBoundary(O, c)`

Input: O: ontology, c: consequence;
Output: boundary b for c;
1. Global `minAs_set= ∅`; (minAs_set is the set of all MinAs for the consequence c)
2. Global `minA:= ∅`; `labelsOfMinA := ∅`; `supremum := ∅`; `infimum := ∅`;
3. `minAs_set := {minA1,......,minAn}`; (get all the minAs of the consequence from the ontology)
4. for each `minAi ∈ minAs_set`, (1 ≤ i ≤ n) do
5. `minAi:={a1,...,am}`; (Get one minA of the consequence from O; a1,...,ai: set of axioms that form the minA)
6. for each `aj ∈ minAi`, (1 ≤ j ≤ m) do
7. `labelsOfMinA:= labelsOfMinA ⨁ label(aj)`;
8. end for
9. `supremum:= supremum ⨁ ⊕ ¦ ¦ label(aj) ∩ minAs_set`;
10. end for
11. `infimum = ⊕ { infimum, supremum }`;
12. `return b`;

Compute HST boundary

The procedure `calculateHSTBoundary(O, c)` takes ontology O and consequence c as input and gives boundary b of c as output. This procedure proceeds as follows, first reasoner checks if O entails c, if it entails then it removes a context `conj` from context lattice (i.e. all the axioms belonging to context) from O and checks if c is entailed. If c is entailed then `conj` is not a minLabel of c, so it is added to nonMinLabs set. If c is not entailed after removing `conj` from O then `conj` is added to minLabels set and axioms of `conj` are added back to O. Calculation of minLabels is continued until whole context lattice is processed. After each iteration of context lattice, we obtain one set of minLabels. If minLabels is not empty then,
supremum \( s_i \) of the \( \text{minLabs} \) obtained and \( s_i \) is added to HST boundary \( b \). After the first iteration we get one set of \( \text{minLabs} \) and algorithm proceeds by removing axioms of one context at a time from the \( \text{minLabs} \) set from the \( O \) and calls the procedure \( \text{calculateHSTBoundary}(O, c) \) again with new set of contexts and axioms of \( O \). This procedure is continued until \( c \) is no more entailed from \( O \). At the stage boundary \( b \) for \( c \) is obtained.

**Algorithm 6** Hitting set tree boundary

**Procedure** \( \text{calculateHSTBoundary}(O, c) \)

**Input:** \( O \): ontology, \( c \): consequence

**Output:** \( b \): boundary of \( c \)

1. Global \( b := \emptyset \), contexts := \{\( c_1, \ldots, c_n \} \); (contexts contains all contexts of the ontology)
2. Global nonMinLabs := \( \emptyset \), minLabels := \( \emptyset \), contextAxioms(\( con, a \)); \( con \): set of contexts \( a \): set of axioms that belong to context \( con \).
   (ContextAxioms contains mapping between contexts and axioms of the ontology)
3. Global count := 0, axiomsProcessedInTree := \( \emptyset \), minLabsRemovedInTree := \( \emptyset \);
   (axiomsProcessedInTree: this is a set variable which contains axioms processed while building the tree for boundary calculation)
4. \( \text{minLabs}_\text{list} := \emptyset \); \( \text{minLabs}_\text{list} \): This list contains minLabels obtained in each iteration
5. if \( O \models c \) then
6.   for each \( con_j \in \text{contextAxioms}(\text{con}, a) \) do
7.     axioms := \( \text{contextAxioms}.\text{get}(\text{con}_j) \);
8.     \( O := O \setminus \text{axioms} \);
9.     if \( O \models c \) then
10.    nonMinLabs := nonMinLabs \cup con_j;
11.   else
12.    minLabels := minLabels \cup con_j;
13.    \( O := O \cup \text{axioms} \);
14. end if
15. end for
16. count := count + 1;
17. else if(\( \text{minLabs}_\text{list} \neq \emptyset \) and \( \text{axiomsProcessedInTree} \neq \emptyset \)) then
18. Local currentMinLab := \( \emptyset \), \( n \);
19. \( \text{minLabs}_\text{list} = \{\text{minLab}_1, \ldots, \text{minLab}_n\} \);
20. axiomsProcessedInTree := \{\( a_1, \ldots, a_m \} \);
21. contextsRemovedInTree := \{\( c_1, \ldots, c_p \} \);
22. currentMinLab := \( \text{minLab}_n \);
23. while(\( \text{currentMinLab} = c_p \)) do
24.   Local currentAxiom := \( a_m \);
25.   axiomsProcessedInTree := axiomsProcessedInTree \setminus a_m;
26.   \( \text{minLabs}_\text{list} = \text{minLabs}_\text{list} \setminus \text{currentMinLab} \);
27.   \( O := O \cup \text{currentMinLab} \);
28. \( n := n - 1 \);
29. \( \text{minLabs}_\text{list} = \{\text{minLab}_1, \ldots, \text{minLab}_n\} \);
30. currentMinLab := \( \text{minLab}_n \);
31. end while
Empirical Evaluation

We have done empirical testing on real-world ontologies of our implementation of the algorithms to

(1) Compute repairs and label ontologies with contexts.
The algorithms to do context based reasoning on the labeled ontologies are

(2) Compute margin-based boundary for the consequence,

(3) Compute HST-boundary for the consequence.
The following sections describe the test data and the test environment first, and then present the empirical results, which show that our algorithms perform well in practical scenarios.
**Test Data and Test Environment**

We performed our tests on a PC with Intel (R) Xeon(R) CPU E5-2640 @ 2.5 GHz and 99143860 KB total memory. We implemented all approaches in Java 1.7 and we used OWL API, as it is convenient for parsing OWL files and to interact with reasoner. We used Hermit reasoner as it offer incremental reasoning.

**Ontologies**

We have chosen the ontologies, which have more minAs for the consequences of the ontology, so that we can efficiently test our implementation.

We used three ontologies from bio-portal, bio ontologies with different expressivities and type of consequences for our experiments. Ontologies used are:

1. Basic Formal Ontology (BFO-1.1.owl),
2. General Formal Ontology (GFO-1.1.owl),
3. Family Health History Ontology (FamilyHealthHistory.owl)

Basic Formal Ontology is a genuine upper ontology, which can be used in support of domain ontologies developed for scientific research, as for example in biomedicine within the framework of the OBO Foundry. This ontology is built using DL ALC. It contains 39 concept names and 277 axioms.

General Formal Ontology is a top-level ontology integrating objects and processes that is built using DL SHIQ. It contains 45 concept names and 479 axioms.

Family Health History Ontology facilitates representing the family health histories of persons related by biological and/or social family relationships. It is built using DL ALCHIF. It contains 238 concept names and 1874 axioms.

**Test setting to assign labels to ontologies**

Computing repairs for consequence of the ontology uses algorithm 2. Assigning labels to axioms of the ontology uses algorithm 1. To assign labels we have chosen various kinds of consequences, like consequences with many repairs, consequences with few repairs, consequences with only one repair and consequences, which are not entailed from the ontology.

**Test setting to compute boundary**

We compute margin-based boundary and Hitting Set Tree (HST) boundary for a consequence, over the context labeled ontology. To compute margin-based boundary algorithm 5 is used. Algorithm 6 is used to compute HST boundary. We have tested different kinds of consequences such as: consequences with many MinAs, consequences with few MinAs, consequences with only one MinA, consequences which do not entail from the ontology, consequences over unlabeled ontologies are also tested.
Experimental Results

Our experiments show that our implementation performs well on large-scale ontologies. In the following we describe our results for each of the two discussed tasks, i.e. labeling the ontologies with contexts and doing context based reasoning over labeled ontologies. Figure 1.1 shows the experimental results of context labeling of ontologies. In Figure 1.1, graph A shows results of BFO-1.1.owl file, graph B shows results of FamilyHealthHistory.owl file and graph C shows results of GFO-1.1.owl file. From the experiments, we analyzed that complexity of labeling the ontology increases as the number of repairs for the consequence increase.

Figures 2.1, 2.2 and 2.3 show experimental results of context-based reasoning done on the labeled ontologies. Figure 2.1 shows the results of BFO-1.1.owl ontology, graphs A, B, C, D are the plots that show Average Time taken to compute HST boundary and Average time taken to compute MB boundary when the number of minAs for the consequences are 2,4,8,16 respectively.

Figure 2.2 shows context-based reasoning results of GFO-1.1.owl ontology. Graphs A, B, C, D, E, F are the plots that show Average Time taken to compute HST boundary and Average time taken to compute MB boundary when the number of minAs for the consequences are 1,3,5,6,7,8 respectively.

Figure 2.3 shows context-based reasoning results of FFH-1.1.owl ontology. Graphs A, B, C, D are the plots that show Average Time taken to compute HST boundary and Average time taken to compute MB boundary when the number of minAs for the consequences are 0, 1, 2, 4 respectively. The case with number of MinAs as 0 is a special case for which consequences are tautologies.

Our results show that Complexity of Margin-Based boundary depends on the reasoner plugged-in. Complexity of Margin-based boundary algorithm grows with the number of minAs for the consequence. Complexity of HST boundary algorithm increases as the number of contexts in the input labeled ontology increase. And As the number of minAs increase HST boundary algorithm performs better than the Margin-based boundary algorithm.

From the results we have learnt that main advantage of Margin-Based boundary algorithm is that any existing implementation can be used to compute all the MinAs. But not all the MinAs may be required to compute the boundary. So computing all the MinAs to compute boundary may be expensive as the number of MinAs grow.

In contrast to this, HST Boundary algorithm do not compute the MinAs, it just computes the MinLabs and not all the MinLabs are required to compute the boundary. So, this algorithm removes redundant labels from the ontology in every step and thus reduces the search space in every step. Instead of computing all the MinAs, this algorithm does intermediate processing to reduce search space which is less complex than computing all MinAs. Thus
for the consequence with more number of MinAs HST-Boundary algorithm performs better than the Margin-Based boundary algorithm.

Experimental Results of Labeling ontologies

Figure 1.1 Context Labeling graphs
Context-Based Reasoning Experimental Results of BFO-1.1.owl

Figure 2.1 Reasoning Graphs of BFO-1.1.owl
Context-Based Reasoning Experimental Results of GFO-1.1.owl

Figure 2.2 Reasoning Graphs of GFO-1.1.owl
Context-Based Reasoning Experimental Results of FamilyHealthHistory.owl

Figure 2.3 Reasoning Graphs of FamilyHealthHistory.owl
Conclusion

In this project we looked into the problem of errors in the owl ontologies and implemented a software tool called COBRA that can deal with this issue. The tool is developed as a two-step process. In the first step we label the ontology to describe contexts, defined as the maximal error-free sub ontologies. In second step is context-based reasoning done on the labeled ontologies, which is the main application of COBRA- Tool. We have tested all our implementations on real world ontologies and results show that our implementations work well on large-scale ontologies.

Reference