An automatic way of generating incoherent terminologies with parameters

Yu Zhang, Dantong Ouyang, and Yuxin Ye *

College of Computer Science and Technology,
Jilin University, Changchun 130012, China

Abstract. The minimal incoherence preserving sub-terminologies (Mips) is defined for identifying the axioms responsible for the unsatisfiable concepts in incoherent ontology. While a great many performance evaluations have been proposed in the past, what remains to be investigated is whether we have effective reasoners to solve the Mips problems, in which case a particular reasoner will be more efficiency than others. After analyzing the structural complexity of terminology, we develop a Mips Benchmark (MipsBM) to evaluate the performances of reasoners by defining six complexity metrics based on concept dependency networks model. Evaluation experiments show that the proposed metrics can effectively reflect the complexity of benchmark data. Not only can the benchmark help the users to determine which reasoner is likely to perform best in their applications, but also help the developers to improve the performances and qualities of their reasoners.

Keywords: Incoherent terminology; Mips; Benchmark; MipsBM

1 Introduction

In practice, building an ontology is a very complicated process and is easy to make errors, an ontology $\mathcal{O}$ is incoherent if there exists an unsatisfiable concept in $\mathcal{O}$, and the existence of unsatisfiable concept indicates that the formal definition is incorrect. Therefore, how to find all the unsatisfiable concepts is the challenging of ontology debugging. Researchers have proposed various methods to debug incoherent ontology. Ontology debugging is achieved by using reasoners, currently, most of the reasoners, such as Pellet [1], HermiT[2], FaCT++[3], TrOWL[4] and JFact support the inference tasks. A great many performance evaluations for reasoners have been performed in the past, What remains to be investigated is whether we have effective reasoners to solve the Mips problems, in which cases a particular reasoner will be more efficiency than others. There are several criteria for a good benchmark test data. First, we need to systematically construct several types of logical contradictions to create an incoherent TBox. Second, there must be a number of parameters that could influence the complexity of benchmark data and the difficulty for reasoning.

* corresponding author: yeyx@jlu.edu.cn (Yuxin Ye).

2 Related Work

In the research of knowledge base query, (LUBM) [5] is developed based on several complexity metrics of ontology and provides 14 test queries to assess the efficiency, correctness and completeness of the knowledge base. However, the correlations between the classes of LUBM are low, thus Li Ma extends it to University Ontology Benchmark (UOBM) [6] by adding a series of association classes. However, either LUBM or UOBM only can evaluate single ontology, thus Yingjie Li et al. [7] develop a multi-ontology synthetic benchmark that can evaluate not only single ontology but also federated ontologies. In the research of ontology matching, Alfio Ferrara et al. [8] propose a disciplined approach to the semi-automatic generation of benchmarks called SWING (Semantic Web Instance Generation), but all the evaluations in SWING are only for single language, so Christian Meilicke et al. [9] design a benchmark for multilingual ontology matching called MultiFarm. Besides, the work in [10] presents the design of a modular test generator to evaluate different matchers on the generated tests. In the research of ontology reasoning and debugging, the benchmarks proposed in [11] and [12] are used to evaluate the classification performances of reasoners. The work in [13] focus on the applicability of specific reasoners to certain expressivity clusters, and evaluate the loading time, classification and conjunctive queries performances of reasoners. JustBench [14] is a typical benchmark to evaluate the reasoners for calcuating justification. In [15], several machine learning techniques are used to predict classification time and determine the metrics that can be used to predict reasoning performance. The work in [16] proposes a method to construct the justification dataset from realistic ontologies with different sizes and expressivities.

3 Complexity Analysis for Incoherent TBox

The expressivity of a particular DL is determined by the concept constructors it provides [17]. $\mathcal{SHOIN}(\mathcal{D})$ is a very expressive DL that provides the constructors including $\mathcal{H}$ (role hierarchies), $\mathcal{O}$(nominals), $\mathcal{I}$(inverse roles), $\mathcal{N}$(Number restriction) and $\mathcal{S}$ is the abbreviation for $\mathcal{ALC}$ with transitive roles. $\mathcal{ALC}$ is the basic description logic consisting of the constructors $\neg C$ (negation), $C \sqcap D$(conjunction), $C \sqcup D$(disjunction), $\exists r.C$(existential restriction) and $\forall r.C$(value restriction).

Example 1. Suppose a $\mathcal{SHOIN}(\mathcal{D})$ TBox contains the following axioms:
$\alpha_1 : S_1 \equiv \exists r_1.B_1$, $\alpha_2 : S_2 \equiv \forall r_2.B_2$, $\alpha_3 : S_3 \equiv \{B_2\} \sqcup \{B_3\}$,
$\alpha_4 : S_4 \sqsubseteq S_1 \sqcup S_2$, $\alpha_5 : S_5 \sqsubseteq \exists r_4.S_3 \sqcap S_4$, $\alpha_6 : S_6 \equiv \exists r_5.B_4$,
$\alpha_7 : S_7 \equiv \{B_4\} \sqcup \{B_5\}$, $\alpha_8 : S_8 \sqsubseteq \exists r_6.S_6 \sqcap S_7$, $\alpha_9 : C_1 \equiv \forall t_1.\neg A_1$,
$\alpha_{10} : C_2 \equiv \exists t_1.A_1 \sqcap \forall t_2.A_2$, $\alpha_{11} : C_3 \equiv \forall t_2.A_2 \sqcap \exists t_2.\neg A_2$, $\alpha_{12} : C_4 \equiv \exists t_2.\neg A_2$,
$\alpha_{13} : C_5 \equiv \geq 3t_3 \sqcap \leq 2t_3$, $\alpha_{14} : C_6 \sqsubseteq C_1 \sqcap C_2$, $\alpha_{15} : C_7 \sqsubseteq C_2$,
$\alpha_{16} : C_8 \sqsubseteq C_4$, $\alpha_{17} : C_9 \sqsubseteq C_7 \sqcap C_8$, $\alpha_{18} : C_{10} \sqsubseteq C_5 \sqcap C_6 \sqcap C_9$,
$\alpha_{19} : r_1 \equiv r_4^-$, $\alpha_{20} : r_2 \sqsubseteq r_3$, $\alpha_{21} : r_5 \circ r_5 \sqsubseteq r_5$

Stefan Schlobach proposes the minimal unsatisfiability preserving sub-TBox (Mups)[18] to identify the axioms responsible for the unsatisfiability of concepts in incoherent TBox. For $\mathcal{T}$ in example 1, it can be shown that the concepts $C_3, C_5, C_6, C_9, C_{10}$ are unsatisfiable by using standard DL TBox reasoning. We can get their Mups:

$\text{Mups}(\mathcal{T}, C_3) = \{\{\alpha_{11}\}\}$, $\text{Mups}(\mathcal{T}, C_5) = \{\{\alpha_{13}\}\}$,
$\text{Mups}(\mathcal{T}, C_6) = \{\{\alpha_9, \alpha_{10}, \alpha_{14}\}\}$, $\text{Mups}(\mathcal{T}, C_9) = \{\{\alpha_{10}, \alpha_{12}, \alpha_{15}, \alpha_{16}, \alpha_{17}\}\}$,
$\text{Mups}(\mathcal{T}, C_{10}) = \{\{\alpha_{13}, \alpha_{18}\}, \{\alpha_9, \alpha_{10}, \alpha_{14}, \alpha_{18}\}, \{\alpha_{10}, \alpha_{12}, \alpha_{15}, \alpha_{16}, \alpha_{17}, \alpha_{18}\}\}$.

Definition 1 (MIPS[18]). *A TBox $\mathcal{T}' \subseteq \mathcal{T}$ is a minimal incoherence preserving sub-TBox* (MIPS) *of $\mathcal{T}$ if and only if $\mathcal{T}'$ is incoherent, and every sub-TBox $\mathcal{T}'' \subset \mathcal{T}'$ is coherent. The set of all* MIPS *of $\mathcal{T}$ is denoted as MIPS($\mathcal{T}$).*

We will abbreviate the set of MIPS for $\mathcal{T}$ by Mips($\mathcal{T}$). For $\mathcal{T}$ in example 1 we can get $\text{Mips}(\mathcal{T}) = \{\{\alpha_{11}\}, \{\alpha_{13}\}, \{\alpha_9, \alpha_{10}, \alpha_{14}\}, \{\alpha_{10}, \alpha_{12}, \alpha_{15}, \alpha_{16}, \alpha_{17}\}\}$.

Definition 2 (Mips Size). *Let Mips($\mathcal{T}$) be the Mips of an incoherent TBox $\mathcal{T}$, the number of axiom set in the Mips($\mathcal{T}$) is called Mips Size.*

Let *Ms* represent the Mips size, for $\text{Mips}(\mathcal{T}) = \{\{\alpha_{11}\}, \{\alpha_{13}\}, \{\alpha_9, \alpha_{10}, \alpha_{14}\}, \{\alpha_{10}, \alpha_{12}, \alpha_{15}, \alpha_{16}, \alpha_{17}\}\}$, there are four axiom sets in the Mips($\mathcal{T}$), thus the Mips size *Ms* =4.

Definition 3 (Mips Depth). *Let Mips($\mathcal{T}$) be the Mips of an incoherent TBox $\mathcal{T}$, the maximum number of axioms in all the axiom sets is called Mips Depth.*

Let *Md* represent the Mips depth. Using the previous example again, both the number of axioms in the first axiom set $\{\alpha_{11}\}$ and the second axiom set $\{\alpha_{13}\}$ are one, while in the third axiom set $\{\alpha_9, \alpha_{10}, \alpha_{14}\}$, the number is three, and in the last axiom set $\{\alpha_{10}, \alpha_{12}, \alpha_{15}, \alpha_{16}, \alpha_{17}\}\}$, the number is five, thus the maximum number of axioms *Md*=5.

Given a TBox $\mathcal{T}$, the concept dependency networks N are defined as follows.

Definition 4 (concept dependency networks). *A directed graph N=(V,E) is a corresponding concept dependency networks of a given TBox $\mathcal{T}$, where V is the set of vertices representing all the concepts in $\mathcal{T}$, and E is the set of edges representing all the dependencies between concepts.*

Figure 1 represents the concept dependency networks of TBox $\mathcal{T}$ in Example 1. On the basis of the concept dependency networks model, the semantic dependency of concept can be defined as follows.

Definition 5 (concept depth). *In the concept dependency networks of TBox $\mathcal{T}$, suppose the concept depth of C is $cd(C)$, $cd(C)$ can be recursively defined as follows.*
if $C \doteq C_1 \sqcap C_2$, then $dep(C) = max(cd(C_1), cd(C_2)) + 1$;
if $C \doteq C_1 \sqcup C_2$, then $dep(C) = max(cd(C_1), cd(C_2)) + 1$;

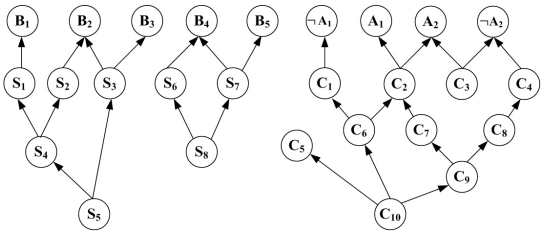


Fig. 1. concept dependency networks of TBox $\mathcal{T}$.

if $C \doteq \exists r.C_1$, *then* $cd(C) = cd(C_1) + 1$;
if $C \doteq \forall r.C_1$, *then* $cd(C) = cd(C_1) + 1$;
if $C \doteq C_1$, *then* $cd(C) = cd(C_1) + 1$;
if $C \doteq \neg C_1$, *then* $dep(C) = cd(C_1) + 1$;
if C *is an atom, then* $cd(C) = 0$;
The $\doteq$ *is either* $\equiv$ *or* $\sqsubseteq$.

If the concept depth of C is 1, C is called a simple concept, otherwise called a complex concept. Suppose that TBox $\mathcal{T}$ contains p simple concepts and q complex concepts, we have the total number of concepts $m = p+q$. Besides, the maximal concept depth of $\mathcal{T}$, denoted as λ, can be defined as: $\lambda = max(cd(C_i)), 1 \leq i \leq m$.

Definition 6 (semantic cluster). *In the TBox* $\mathcal{T}$, *the subTBox* $\mathcal{T}' \subseteq \mathcal{T}$ *which is composed of concepts linked together by semantic dependency relationship, is called a semantic cluster of* $\mathcal{T}$.

Suppose that the number of semantic dependency is μ. The semantic cluster must satisfy the constraint $p + \mu \sum_{i=1}^{\lambda} dep(C_i) = m$. Furthermore, the clustering coefficient can be defined as:

$$\eta = \frac{\mu \sum_{i=1}^{\lambda} dep(C_i)}{m}. \tag{1}$$

If $\mu = 0$, which means there is not any semantic cluster in the TBox, so the minimum of clustering coefficient $\eta_{min} = 0$. If, however, $p = 0$, which means the TBox is composed only of complex concepts, then $\mu \sum_{i=1}^{\lambda} dep(C_i) = m$, so the maximum of clustering coefficient $\eta_{max} = 1$.

4 MipsBM System

MipsBM consists of two components: satisfiable concept generator and unsatisfiable concept generator. According to the characteristics of axioms appearing in $\mathcal{SHOIN}(\mathcal{D})$ TBox, we categorize them into two groups: constructors and operands. The constructors group consists of concept constructor and property constructor. And the operands group is composed of atom set and role set. The constructors and operands table are shown in Table 1.

Table 1. Constructors and Operands Table

Satisfiable Constructors		Unsatisfiable Constructors	
SatConceptConstructor	Syntax	unSatConceptConstructor	Syntax
subClass	$S_1 \sqsubseteq S_2$	subClass	$C_1 \sqsubseteq C_2$
equivalentClass	$S_1 \equiv S_2$	equivalentClass	$C_1 \equiv C_2$
intersection	$S_1 \sqcap S_2$	intersection	$C1 \sqcap C_2$
allValues	$\forall r_1.B_1$	allValues	$\forall t_1.A_1$
someValues	$\exists r_2.B_2$	someValues	$\exists t_2.A_2$
union	$S_1 \sqcup S_2$	complement	$\neg A$
oneOf	$\{x_1, x_2, x_3\}$	disjointWith	$C_1 \sqsubseteq \neg C_2$
PropertyConstructor	Syntax	maxCardinality	$\geq nr_1$
subProperty	$r_1 \sqsubseteq r_2$	minCardinality	$\leq nr_1$
equivalentProperty	$r_1 \equiv r_2$	PropertyConstructor	Syntax
TransitiveProperty	$r_1^+ \sqsubseteq r_2$	subProperty	$t_1 \sqsubseteq t_2$
inverseOf	r^-	equivalentProperty	$t_1 \equiv t_2$
Satisfiable Operands		Unsatisfiable Operands	
SatAtomSet	$B_1, B_2, \cdots, B_m$	unSatAtomSet	$A_1, A_2, \cdots, A_n$
SatRoleSet	$r_1, r_2, \cdots, r_m$	unSatRoleSet	$t_1, t_2, \cdots, t_n$

Algorithm 1: satGenerator

Input: *satnum*: number of satisfiable concepts
μ: number of semantic clusters
λ: maximum concept depth
output: S: satisfiable concept set

while$(\mu > 0)$
  $constructor_k = randomSelect(SatConceptConstructor)$;
  $op_k = randomSelect(SatAtomSet,SatRoleSet)$;
  $S_k = generateAxiom(constructor_k, op_k), S.add(S_k), k++$;
  while$(\lambda > 0$)
    $conceptstructor_k = randomSelect(SatConceptConstructor)$;
    $propstructor_k = randomSelect(SatPropertyConstructor)$;
    $op_k = randomSelect(SatAtomSet,SatRoleSet), op_k = op_k \cup S_k$;
    $\{S_k, P_k\} = generateAxioms(conceptstructor_k, propstructor_k, op_k)$;
    $S = S \cup \{S_k\}, \lambda --, k++$;
  $\mu --$;
$num = satnum - size(S)$;
while$(num \geq 0)$
  $constructor_k = randomSelect(SatConceptConstructor)$;
  $op_k = randomSelect(SatAtomSet,SatRoleSet)$;
  $S_k = generateAxiom(constructor_k, op_k)$;
return S

The proof for Algorithm 1 is as follows.

Proof. Because there are not any complement or disjoint constructors in the Satisfiable Constructors in Table 1, the concepts generated by Algorithm 1 must be satisfiable.

The first while loop corresponds to the number of semantic clusters, in each loop, the algorithm creates a semantic cluster, and the value of μ is decreased by 1 until $\mu = 0$. The second while loop corresponds to the maximum concept depth, in each loop, the algorithm creates a concept, and the concept depth of the latter concept is 1 bigger than that of the former one. When the loop is finished, the concept depth of the last concept reaches λ. After that, the number of satisfiable concepts is obtained, the rest of the concepts are created in the third while loop.

In order to build an incoherent terminology, MipsBM needs to create several unsatisfiable concepts which can be achieved through systematically constructing logical clashes.

Definition 7 (Independent Unsatisfiable Concept). *C is an independent unsatisfiable concept if the unsatisfiability of C depends on the concept definition rather than the unsatisfiability of other concepts.*

Definition 8 (Dependent Unsatisfiable Concept). *C is a dependent unsatisfiable concept if the unsatisfiability of C depends on the unsatisfiability of other concepts.*

From the Example 1, C_3, C_5, C_6 and C_9 are independent unsatisfiable concepts, C_{10} is dependent unsatisfiable concept because its unsatisfiability depends on unsatisfiable concepts C_5, C_6, C_9.

Definition 9 (Clash Sequences). *Let $Seq^+(C)$ be the positive clash sequence of C, and $Seq^-(C)$ the negative clash sequence. $Seq^+(C)$ is of the form $< (C_1, I_1, C_2), (C_2, I_2, C_3), \cdots, (C_m, I_m, C) > (i = 1, \cdots, m)$, where $C_i \sqsubseteq C_{i-1}$, I_i represents the indexes of axioms related to $C_i \sqsubseteq C_{i-1}$. $Seq^-(C)$ is of the form $< (\neg C_1, I'_1, C'_2), (C'_2, I'_2, C'_3), \cdots, (C'_n, I'_n, C) > (i = 1, \cdots, n)$, where $C'_i \sqsubseteq C'_{i-1}$, I'_i represents the indexes of axioms related to $C'_i \sqsubseteq C'_{i-1}$. After that, the unsatisfiable concept C can be generated by $C \sqsubseteq C_m \sqcap C'_n$.*

For example, The clash sequences of C_9:

$Seq^+(C_9) = < (A_1, \{\alpha_{10}\}, C_2), (C_2, \{\alpha_{10}, \alpha_{15}\}, C_7), (C_7, \{\alpha_{10}, \alpha_{15}, \alpha_{17}\}, C_9)) >$,

$Seq^-(C_9) = < (\neg A_1, \{\alpha_{12}\}, C_4), (C_4, \{\alpha_{12}, \alpha_{16}\}, C_8), (C_8, \{\alpha_{12}, \alpha_{16}, \alpha_{17}\}, C_9) >$.

Unsatisfiable concepts can be divided into two types as follows.

complement clash: C is a *complement clash* concept if it is a subclass of both class A and the complement of class A. For example:

$\alpha_1 : C_1 \sqsubseteq \forall t_1.A_1 \sqcap \exists t_1.\neg A_1$. Then C_1 is a *complement clash* root concept.

cardinality clash: C is a *cardinality clash* concept if the at-least restriction is bigger than the at-most restriction in its definition. For example:

$\alpha_2 : C_2 \equiv \geq 2.t_2 \sqcap \leq 1.t_2$. Then C_2 is a *cardinality clash* root concept.

Unsatisfiable concept generator (Algorithm 2) creates the satisfiable concepts by constructing clash sequences.

Algorithm 2: unsatGenerator(*unsatnum*,*Ms*,*iMd*)

inputs:*unsatnum*: number of unsatisfiable concept
Ms: Mips size; *iMd*: increasement of Mips depth
output: U: unsatisfiable concept set; Mips($\mathcal{T}$): the Mips of $\mathcal{T}$

01 $U = \varnothing$, Mips($\mathcal{T}$) $= \varnothing, k = 0, len = 0$;
02 *constructor = randomSelect(UnsatConceptConstructor;)*
03 construct a pair of clsh sequences : $\{Seq^+, Seq^+\}$
04 $D_0 \leftarrow Seq^+$, $D'_0 \leftarrow Seq^-$;
05 $I_{(C_k)} : C_k \doteq intersectionOf(D_0, D'_0)$;
06 $C_R.add(C_k)$, $Mips.add(I_{(C_k)})$, k++, *len*++;
07 **while**($k \leq Ms$)
08 *len=len+iMd*;
09 construct a pair of clsh sequences : $\{Seq^+, Seq^+\}$
10 $D_0 \leftarrow Seq^+$, $D'_0 \leftarrow Seq^-$;
11 **for**(*i=j=1*; $j <$ *len*; *i++,j=j+2*)
12 $S_{x,y} \leftarrow$ *(SatAtomSet,SatRoleSet,someValues,allValues)*;
13 $I_{(D_i)} : D_i \doteq intersectionOf(D_{i-1}, S_x)$;
14 $I'_{(D'_i)} : D'_i \doteq intersectionOf(D'_{i-1}, S_y)$;
15 $Mips.add(I_{(D_i)}, I'_{(D'_i)})$;
16 $I_{(C_k)} : C_k \doteq intersectionOf(D_i, D'_i)$, $C_R.add(C_k)$, $Mips.add(I_{(C_k)})$;
17 U.add(C_R), Mips($\mathcal{T}$).add(*Mips*), k++;
18 $num = unsatnum - k$;
19 **while**($m \leq num$)
20 $C_r \leftarrow$*randomSelect*(C_R);
21 $S_z \leftarrow$ *(SatAtomSet,SatRoleSet,someValues,allValues)*;
22 $C_k \doteq intersectionOf(C_r, S_z)$;
23 U.add(C_k),$m++$;
24 **return** $U, Mips(\mathcal{T})$

Theorem 1 *The unions of clash sequences of independent unsatisfiable concepts are the Mips of TBox.*

Proof. By Definition 1(Incoherent TBox), we have that a TBox $\mathcal{T}$ is incoherent if and only if there is a concept name in $\mathcal{T}$ which is unsatisfiable. Therefore, according to Definition 3(Mips), we can prove Theorem 1 based on two points:

■ One concept is unsatisfiable in the union of contradiction sequences.

■ And the concept is satisfiable in every subset of the union of contradiction sequences.

We prove the first point. Let C_k be a satisfiable concept, According to the unsatGenerator algorithm, C_k is created by $C_k \sqsubseteq D_i \sqcap D'_i$, where $D_i \sqsubseteq D_{i-1}$ and $D'_i \sqsubseteq D'_{i-1}$. Similarly, $D_{i-1} \sqsubseteq D_{i-2}, \cdots, D_2 \sqsubseteq D_1$ and $D'_{i-1} \sqsubseteq D'_{i-2}, \cdots, D'_2 \sqsubseteq D'_1$. The corresponding clash sequences are:

$< (D_1, I_1, D_2), (D_2, I_2, D_3), \cdots, (D_i, I_i, C_k) >$, where $I_i = I_i \cup I_{i-1}$.

$< (D'_1, I'_1, D'_2), (D'_2, I'_2, D'_3), \cdots, (D'_i, I'_i, C_k) >$, where $I'_i = I'_i \cup I'_{i-1}$.

D_1 and D'_1 have the form either $D_1 \equiv A, D'_1 \equiv \neg A$ or $D_1 \equiv \geq mt, D'_1 \equiv \leq nt (m > n$, and t is a role name). this implies that $C_k \sqsubseteq D_1$ and $C_k \sqsubseteq D'_1$, i.e.

$C_k \sqsubseteq A \sqcap \neg A$ or $C_k \sqsubseteq \geq mt\sqcap \leq nt(m > n)$. Therefore, C_k is unsatisfiable in $\mathcal{T}' = I_i \cup I'_i$, i.e. C_k is unsatisfiable in the union of clash sequences.

Next, we prove the second point. Let $\mathcal{T}''$ be the every subset of $\mathcal{T}'$ after removing any one axiom α_j from $I_i \cup I'_i$. If α_j occurs in the Seq^+ of C_k, we have that $D_i \sqcap D_{i-1}, D_{i-1} \sqsubseteq D_{i-2}, \cdots, \alpha_j : D_j \sqsubseteq D_{j-1}, \cdots, D_2 \sqsubseteq D_1$. Removing α_j is equivalent to removing $D_j \sqsubseteq D_{j-1}$ from the Seq$^+$ of C_k, so D_i is not the subset of D_1. If α_j occurs in the Seq^- of C_k, we have that $D'_i \sqcap D'_{i-1}, D'_{i-1} \sqsubseteq D'_{i-2}, \cdots, \alpha_j : D'_j \sqsubseteq D'_{j-1}, \cdots, D'_2 \sqsubseteq D'_1$. Removing α_j is equivalent to removing $D'_j \sqsubseteq D'_{j-1}$ from the Seq$^-$ of C_k, so D'_i is not the subset of D'_1. We know $C_k \sqsubseteq D_i \sqcap D'_i$, so C_k is not the subset of both D_1 and D'_1. Therefore, C_k is satisfiable in $\mathcal{T}''$, i.e. C_k is satisfiable in every subset of the union of clash sequences.

5 Evaluation with MipsBM

The MipsBM experiments demonstrate how to evaluate the performances of reasoners for Calculating Mips. Pellet 2.3.1 [1], HermiT 1.3.8 [2], FaCT++ 1.6.2 [3], JFact 1.0.0 [4] and TrOWL 1.4 [5] are the five most widely-used description logics reasoners used in our experiments. The tests are performed on a PC (Intel(R) Core(TM) CPU 3.40Ghz) with 4 GB RAM. Our performance measure is the run time (in seconds) to calculate Mips.

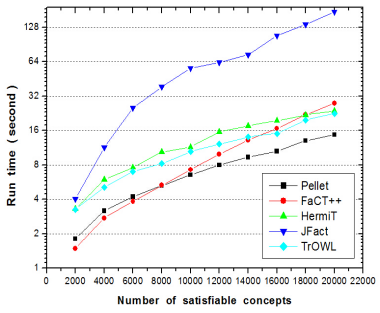


Fig. 2. evaluations for the number of satisfiable concepts

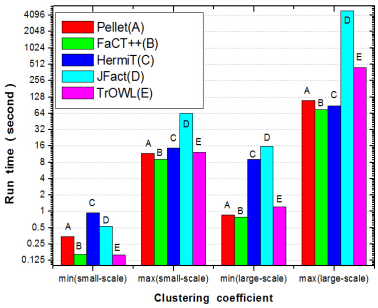
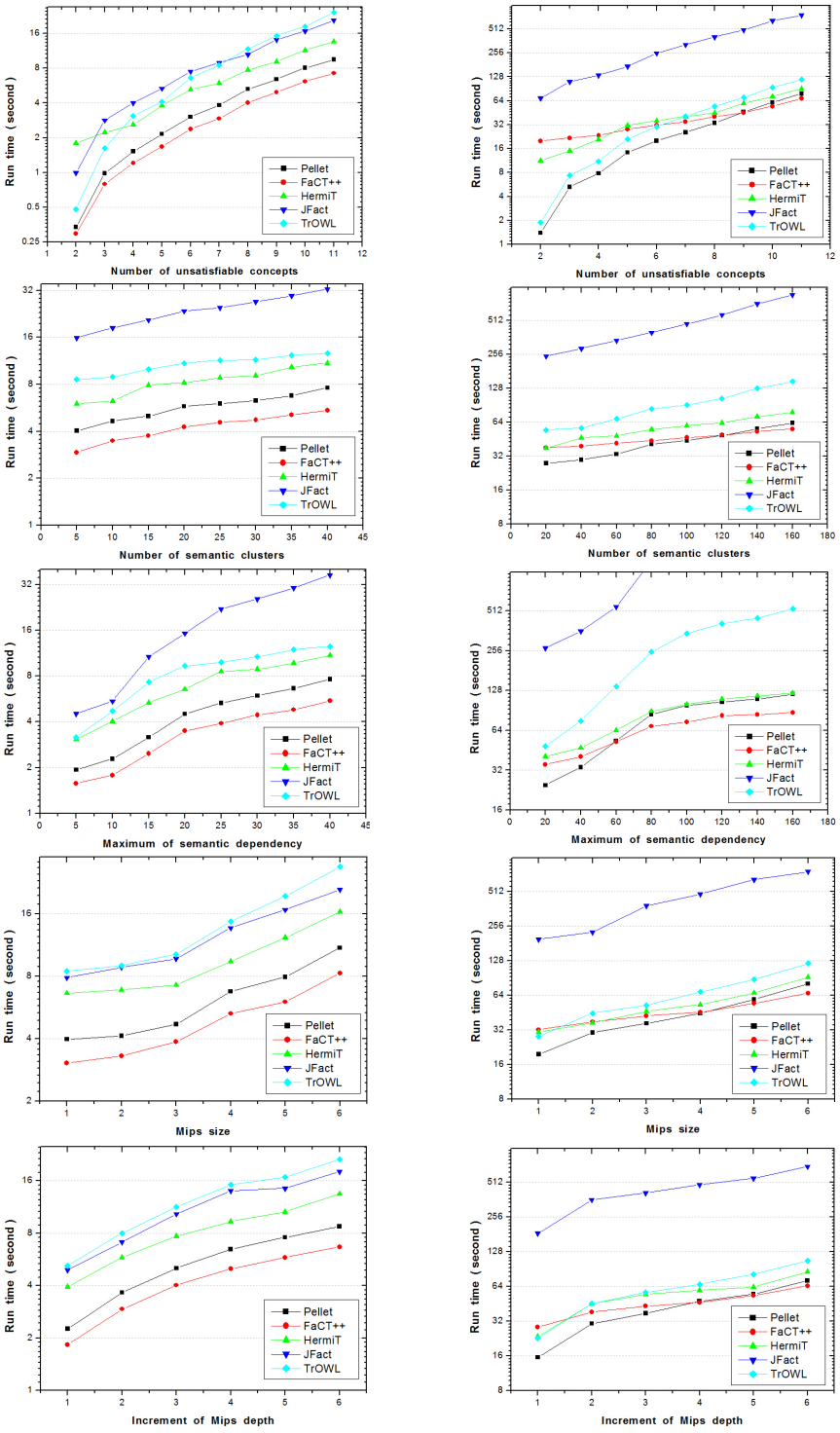


Fig. 3. evaluations for the clustering coefficient

From Figure 2, we can conclude from the second experiment that TBox size plays a significant influence on the performances of different reasoners. Therefore, the following evaluations can be viewed from two aspects: small scale TBox (the number of concepts $m = 2000$) and large scale TBox ($m = 20000$).

[1] http://clarkparsia.com/pellet
[2] http://www.hermit-reasoner.com/
[3] http://code.google.com/p/factplusplus/
[4] http://sourceforge.net/projects/jfact/
[5] http://trowl.eu/



In the case of large scale TBox

In the case of small scale TBox

Fig. 4. performance evaluations for reasoners about complexity metrics

After the evaluation experiments, we give a further analysis from two perspectives.

What makes an incoherent TBox difficult to calculate Mips? In order to answer this question, we consider the impact of construction parameters on structure complexity of incoherent terminology. A large number of satisfiable concepts mean a large size of TBox, Reasoners have to take a lot of time to perform satisfiability checking, so the run time becomes longer. There are many relevance relations between one concept and others if the concept depth is large, as the number of semantic clusters increases, the number of semantic dependencies between the concepts will grow significantly. The Mips size corresponds to the scale of minimal conflict axiom set, our reasoners need to find the minimal conflict axiom set of the incoherent TBox, thus the size of semantic dependency is strictly determined by the Mips depth. According to Definition 9, the clash sequences of unsatisfiable concepts correspond to the increase of Mips depth, the larger the depth is, the longer the clash sequences are, therefore, a larger value of the increase of Mips depth leads to a higher complex of incoherent TBox.

Which is the most appropriate reasoner to solve Mips problem? Because of the differences of optimization approaches, the five reasoners have different performances for the same benchmark test data. When the number reaches 8000, Pellet is faster than FaCT++, when reaches 14000, TrOWL is faster than FaCT++, and when reaches 18000, HermiT performs better than FaCT++. In the process of consistency checking, HermiT uses the anywhere blocking technique to limit the sizes of models which are constructed, so it has an advantage over ABox. Unfortunately, the ontology test data generated by our MipsBM only consists of TBox, thus the advantages haven't been fully fulfilled. Our experiments show that timeout is the main reason to cause the failures of JFact, especially for a large inputs, It is because JFact takes longer to load the TBox than others. In the case of large scale TBox, JFact fails to resolve the Mips problems when the number of clusters increases beyond 80 in the fourth experiment.

6 Conclusion and future work

This paper presents a benchmark to generate different complicated terminologies to evaluate the performances of description logics reasoners for calculating Mips. Our purpose is to find out the reasons which result in the difficulty and high cost of ontology debugging. Experiments show that the six construction parameters can fully reflect the complexity of incoherent TBox.

As for future work, we plan to improve our benchmark under realistic semantic web conditions to evaluate reasoners by using realistic TBox data, and focus on different ontology reasoning and debugging algorithms to evaluate their completeness and correctness by using our extended benchmark.

References

1. Sirin Evren, Parsia Bijan, et al., Pellet:A practical OWL-DL reasoner, Web Semantics: science, services and agents on the World Wide Web, 2007, 5(2):51-53.

2. Rob Shearer, Boris Motik, and Ian Horrocks, HermiT: A highly-efficient owl reasoner, in: Proceedings of the 5th International Workshop on OWL: Experiences and Directions. Karlsruhe, Germany. 2008.
3. Dmitry Tsarkov and Ian Horrocks, FaCT++ Description Logic Reasoner: System Description, in: Proceddings of Third International Joint Conference on Automated Reasoning. Seattle, WA, USA, 2006, pp.292-297.
4. Edward Thomas, Jeff Z. Pan, Yuan Ren, TrOWL: Tractable OWL 2 Reasoning Infrastructure, in: Proceedings of 7th Extended Semantic Web Conference. Heraklion, Crete, Greece, 2010, pp.431-435.
5. Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin, LUBM: A benchmark for OWL knowledge base systems, Web Semantics: Science, Services and Agents on the World Wide Web, 2005, 3(2): 158-182.
6. Li Ma, Yang Yang, Zhaoming Qiu, et al., Towards a complete owl ontology benchmark, in: Proceedings of the 3rd European Semantic Web Conference (ESWC), Budva, Montenegro, June, 2006, pp.125-139.
7. Yingjie Li, Yang Yu and Jeff Hefli, A multi-ontology synthetic benchmark for the semantic web, in: Proceedings of the 1st International Workshop on Evaluation of Semantic Technologies, Shanghai, China. 2010.
8. Alfio Ferrara, Stefano Montanelli, et al., Benchmarking matching applications on the semantic web, The Semanic Web: Research and Applications, Springer Berlin Heidelberg, 2011, pp.108-122.
9. Christian Meilicke, Raul Garca-Castro, et al., MultiFarm: A benchmark for multilingual ontology matching, Web Semantics: Science, Services and Agents on the World Wide Web, 2012, 15: 62-68.
10. Maria Rosoiu, Cassia Trojahn Dos Santos, et al., Ontology matching benchmarks: generation and evaluation, in: Proceedings of the 6th ISWC workshop on ontology matching (OM), 2011, pp.73-84.
11. Zhengxiang Pan, Benchmarking DL Reasoners Using Realistic Ontologies, in: Proceedings of the OWLED05 Workshop on OWL: Experiences and Directions, Galway, Ireland, November, 2005, 188.
12. Tom Gardiner, Ian Horrocks and Dmitry Tsarkov, Automated benchmarking of description logic reasoners, in: Proceedings of the 19th International Workshop on Description Logics, Windermere, Lake District, UK, May, 2006, pp. 167-174.
13. Jurgen Bock, Peter Haase, et al., Benchmarking OWL reasoners, in: Proceedings of the ARea 2008 Workshop, Tenerife, Spain, June, 2008.
14. Samantha Bail, Bijan Parsia, Ulrike Sattler, JustBench: a framework for OWL benchmarking, in: Proceedings of the 9th International Semantic Web Conference (ISWC), Shanghai, China, November, 2010, pp.32-47.
15. Yong-Bin Kang, Yuan-Fang Li, Shonali Krishnaswamy, Predicting reasoning performance using ontology metrics, in: Proceedings of the 11th International Semantic Web Conference (ISWC), Boston, MA, USA, November, 2012, pp.198-214.
16. Ji Qiu, Gao Zhiqiang, Huang Zhisheng, et al., Measuring effectiveness of ontology debugging systems, Knowledge-Based Systems, 2014, 71: 169-186.
17. Kathrin Dentler, Ronald Cornet, et al., Comparison of reasoners for large ontologies in the OWL 2 EL profile, Semantic Web, 2011, 2(2): 71-87.
18. Stefan Schlobach, Zhisheng Huang, Ronald Cornet, et al., Debugging incoherent terminologies, Journal of Automated Reasoning, 2007, 39(3):317-349.
19. Aditya Kalyanpur, Debugging and repair of owl ontologies, Washington DC, American: The University of Maryland, 2006.