Jurnal Teknik Informatika (JUTIF)
Vol. 5, No. 4, August 2024, pp. 513-520

DOI: https://doi.org/10.52436/1.jutif.2024.5.4.2089
p-ISSN: 2723-3863
e-ISSN: 2723-3871

THE EVALUATIONS FOR THE BACKEND OF ONTI MEASURES WITH BLACK BOX METHOD

Nur Alfi Ekowati*[1], Sulistiyasni[2], Ika Indah Lestari[3]

[1]Informatics, Faculty of Engineering, Universitas Jenderal Soedirman, Indonesia
[2,3]Informatics Engineering, STMIK Widya Utama, Indonesia
Email: [1]nuralfi.ekowati@unsoed.ac.id, [2]sulistiyasnipwt@swu.ac.id, [3]ikaindah22@swu.ac.id

(Article received: May 12, 2024; Revision: July 18, 2024; published: August 14, 2024)

Abstract

Inconsistency in an ontology can be a serious problem since it can mess up the information in the ontology. Ontology-based inconsistency measure gives inconsistency value of the whole base of the OWL ontology. It means the produced inconsistency value is used to evaluate its whole base. Based on this characteristic, there were 10 inconsistency measures created in the previous research and collected into one package of measures in an application program, namely Onti Measures. The application will not be useful if the measures do not work well. This problem leads to conduct evaluations. In this research, evaluations for the backend part of Onti Measures with the use of three kinds of OWL reasoners are done to know the performance of the application system with the comparison of each reasoner usage. The evaluations for the whole part of the application are not the scope of this research since they are only done for the backend part. Particularly, they are done with the black box method since the structure of the codes are not necessary to be known. They are evaluated with several OWL files as test cases and as the inputs of the backend program. The evaluation shows that the same inconsistent OWL file that is computed with a different type of inconsistency measure with any chosen reasoner may result in different inconsistency value. Other evaluations are provided. Overall, they show that Pellet is better than the two other reasoners and I_(D_f) is more efficient than the other measures.

Keywords: *backend, inconsistency measures, Onti Measures, OWL ontologies, OWL reasoner*.

1. INTRODUCTION

Description Logics are a family of knowledge representation languages that are widely used in ontological modeling [1]. In general, ontology is usually related to an important role in the semantic web. One cannot deny if there is always possibility that an inconsistent knowledge occurs in the ontology. For example, if it is compiled by more than one person. High inconsistency can be a serious problem that needs to be solved since it can mess up the information in the ontology. Inconsistency measures are necessary in such case.

Inconsistency measures for ontology are useful to analyze the inconsistency of the ontologies. Then it can give insights to handle the inconsistencies. Many inconsistency measures have been created for Propositional Logic, such as the ones that are proposed by [2], [3], [4], [5]. Some inconsistency measures for First-Order Logic have been created as well, such as the ones that are proposed by [6], [7]. On the other hand, inconsistency measures for OWL ontologies with axiom-based computations have been built by [8]. In contrast to the axiom-based inconsistency measure that defines the inconsistency for each axiom of the ontology, ontology-based inconsistency measure gives inconsistency of the whole base of ontology, i.e., to evaluate whole base.

Onti Measures is a name that is created to call a package of 10 ontology-based inconsistency measures, as mentioned in Table 1. Each of them was previously defined and applied in some examples of written implementation in [8]. Most of the measures were created by transferring or converting the inconsistency measures for Propositional Logic in the survey of [5] which includes [9], [10], [11], [12], into measures for ontology language OWL 2. The ones that are not from the surveys were created by transferring them as well.

Onti Measures had been built in the form of a web application and had been introduced in [13]. The descriptive analysis of test data using three reasoners that are embedded in the Onti Measures had been analyzed in detail by [14]. In general, brief evaluations of Onti Measures for the whole part of the application, i.e., backend and frontend parts were done by [13], [14], while evaluations of each part of them had not been done yet. The application will not be useful if the measures do not work well. This problem leads to conduct evaluations. That is the importance of this research which is aimed at evaluating the backend part of the Onti Measures with the black box method and with the use of three kinds of OWL reasoners to know the performance. Based on the research [15], [16] the comparisons of some OWL reasoners have existed. However, they are not related to the inconsistency measures that are in the Onti Measures.

Table 1. Inconsistency Measures in Onti Measures

Onti Measures		
Category	**Inconsistency Measure Name**	**Initial Name (Symbol)**
Drastic inconsistency measures	Drastic Inconsistency Measure	$\mathcal{I}_\mathrm{d}$
Minimal inconsistency-based measures	MI-inconsistency Measure	$\mathcal{I}_\mathrm{MI}$
	MI^C-inconsistency Measure	$\mathcal{I}_{\mathrm{MI}^\mathrm{C}}$
	D_f-inconsistency Measure	$\mathcal{I}_{\mathrm{D}_f}$
	Problematic Inconsistency Measure	$\mathcal{I}_\mathrm{p}$
	Incompatibility Ratio Inconsistency Measure	$\mathcal{I}_\mathrm{IR}$
Maximal consistency-based measures	MC-inconsistency Measure	$\mathcal{I}_\mathrm{mc}$
	The nc-inconsistency Measure	$\mathcal{I}_\mathrm{nc}$
Variable-based measures	The mv-inconsistency Measure	$\mathcal{I}_\mathrm{mv}$
	ID_MCS Inconsistency Measure	$\mathcal{I}\mathrm{D}_\mathrm{MCS}$

2. METHOD

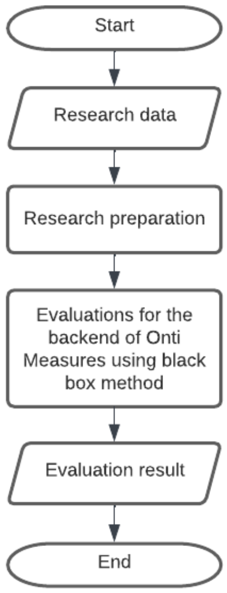
In the scope of this research the evaluations of Onti Measures with the user interface (frontend part) are not applicable. Instead, the evaluations are only done for the backend part of the program. The backend part of Onti Measures is written with the Java language and with the use of OWL files as the real input. It is run in Eclipse Java Neon with the use of Java SE Development Kit 8. The usage of the program depends on the IDE (Integrated Development Environment), i.e., the Eclipse. This means one should open the IDE to run the program.

The implementation needs some libraries to support the running of the program. Some of them are OWL API that can create and manipulate the OWL ontologies; and OWL explanation that is able to retrieve the minimal subset of the ontology related to the entailment to hold. The usage of OWL reasoner plays a substantial role in the implementation of the measures. It can give services which have prominent advantages in using ontologies, such as consistency. HermiT, JFact, and Pellet are used as the OWL reasoners of the program. The reasoners are chosen because they are the easiest ones to integrate with the application system. The performance of the measures will be compared with different OWL reasoners.

Figure 1 depicts the steps of this research. The research data is as the input of the application, that is a number of OWL ontologies. Research preparation is to prepare all the things that are needed to evaluation, such us the Onti Measures. The evaluations of the program are done by doing experiments with the black box testing method. Black box testing is also called a functional testing technique. Black box testing does not concern with the internal mechanisms of a system, instead it focuses solely on the outputs generated in response to selected inputs and execution conditions [17]. The code is purely considered to be a "big black box" to the tester who cannot see inside the box [18]. In this research, the evaluations by doing the testing are not done by the one dedicated only as a tester. Instead, it is done by the software developer who works as a tester as well, in which those positions are taken by the first author of this research.

The testing of the main function is done by employing 36 OWL files as the test cases and as the inputs mentioned earlier. They are obtained from a website related to semantic website [19] in which all of them are claimed as inconsistent ontologies. The test cases are included as DL SROIQ. Most of mainstream DLs today are, in fact, sublanguages of DL SROIQ [20].

The step to do the black box testing in this research is started by opening the IDE and choosing the workspace folder where the project of Onti Measures is located on the computer. After that, the OWL files as the input of the program should be placed in the data folder in the project explorer of the IDE. Choose a reasoner among the provided three reasoners, and run the application is the next step. Once Onti Measures has finished running, the inconsistency values are generated in the output folder. Since Onti Measures consists of 10 measures, it should generate 10 values in files as the results.


Start
Research data
Research preparation
Evaluations for the backend of Onti Measures using black box method
Evaluation result
End
Figure 1. The Steps of the Research

3. RESULTS

There are 36 OWL ontologies in Tabel 2 which are used as the test cases of the evaluations.

Table 2. Order of Ontology Test Cases

Order	Ontology Name (*.owl)
1	DisjointClasses-002
2	New-Feature-AsymmetricProperty-001
3	New-Feature-BottomObjectProperty-001
4	New-Feature-IrreflflexiveProperty-001
5	New-Feature-NegativeObjectPropertyAssertion-001
6	New-Feature-TopObjectProperty-001
7	Rdfbased-sem-bool-complement-inst
8	Rdfbased-sem-char-asymmetric-inst
9	Rdfbased-sem-char-asymmetric-term
10	Rdfbased-sem-char-irreflflexive-inst
11	Rdfbased-sem-class-nothing-ext
12	Rdfbased-sem-eqdis-different-sameas
13	Rdfbased-sem-eqdis-disclass-eqclass
14	Rdfbased-sem-eqdis-disclass-inst
15	Rdfbased-sem-ndis-alldifferent-fw
16	Rdfbased-sem-ndis-alldifferent-fw-distinctmembers
17	Rdfbased-sem-ndis-alldisjointclasses-fw
18	WebOnt-Nothing-001
19	WebOnt-Restriction-001
20	WebOnt-Restriction-002
21	WebOnt-Thing-003
22	WebOnt-description-logic-001
23	WebOnt-description-logic-002
24	WebOnt-description-logic-003
25	WebOnt-description-logic-010
26	WebOnt-description-logic-011
27	WebOnt-description-logic-012
28	WebOnt-description-logic-013
29	WebOnt-description-logic-032
30	WebOnt-description-logic-033
31	WebOnt-description-logic-040
32	WebOnt-description-logic-101
33	WebOnt-description-logic-102
34	WebOnt-description-logic-103
35	WebOnt-description-logic-104
36	WebOnt-description-logic-110

Besides the 36 test cases which contain inconsistent ontologies, the three OWL reasoners are also employed in the program, i.e., HermiT, JFact, and Pellet. Hence, the usage of Onti Measures along with the three reasoners should be compared to know the performance of them all. The evaluations are discussed in the following explanations.

3.1. Inconsistency Value (wrt. the Reasoner)

Some charts will depict inconsistency values obtained by $\mathcal{I}_{\mathrm{d}}$, $\mathcal{I}_{\mathrm{mc}}$, and $\mathcal{I}_{\mathrm{mv}}$ as the representative measures of Onti Measures to show here, with the use of the reasoners. The $\mathcal{I}_{\mathrm{d}}$ is shown to represent the simplest computation. In Figure 2 the X axis in the chart represents the order of ontology test cases in which the names of OWL ontology files are as shown in Table 2.

Each number of ontologies in the order has three bars in three colors in which orange is for the inconsistency measurement of the related ontology with the reasoner HermiT. The yellow and green ones are for the computation with the reasoner JFact and Pellet, respectively. One example to read the chart is ontology number 1 (DisjointClasses-002) which has been computed with inconsistency measure $\mathcal{I}_{\mathrm{d}}$ in Onti Measures program along with reasoner HermiT has given result 1. The same result happens when it was computed either with JFact or with Pellet. Since all ontology is claimed to be inconsistent, other ontologies should have values 1.

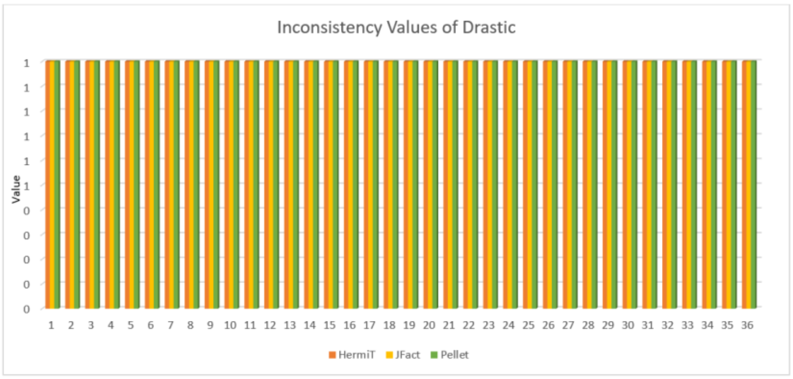


Figure 2. Inconsistency Values Obtained by $\mathcal{I}_{\mathrm{d}}$

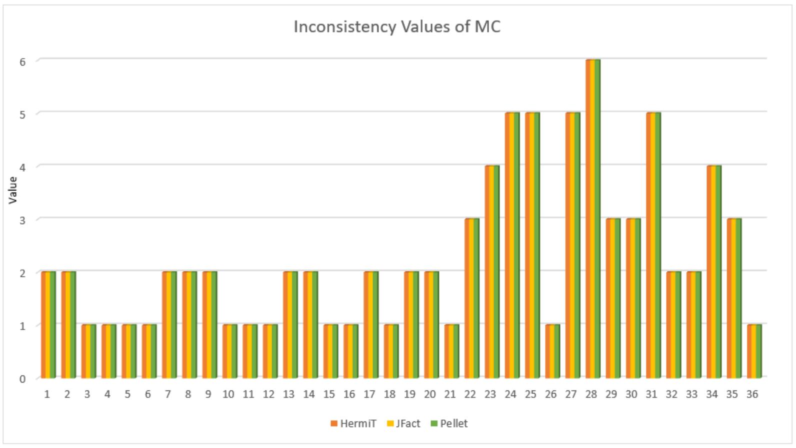


Figure 3. Inconsistency Values Obtained by $\mathcal{I}_{\mathrm{mc}}$

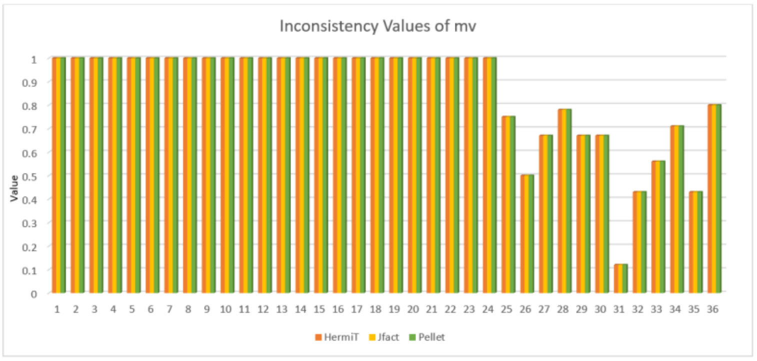


Figure 4. Inconsistency Values Obtained by $\mathcal{I}_{\mathrm{mv}}$

By the chart, it is shown that the measure $\mathcal{I}_{\mathrm{d}}$ along with the three reasoners HermiT, JFact, and Pellet work correctly to help compute the inconsistency degrees of the test cases. The same conditions happen to the measure $\mathcal{I}_{\mathrm{mc}}$ and $\mathcal{I}_{\mathrm{mv}}$, which are shown in Figure 3 and Figure 4 to represent the middle ones, i.e., not the simplest nor the most complicated one in terms of the simplicity of the calculation formula.

The usage of the measure along with any reasoner can work well and give the same result in terms of resulting the inconsistency value wrt. any used reasoner. This evaluation claims that the same inconsistent OWL file that is computed with different measure may result in the different inconsistency value, but any chosen OWL reasoner does not impact the difference.

3.2. Inconsistency Value Quantity

The quantities of inconsistency values obtained by the measures in the package of Onti Measures are presented in Table 3 and Table 4. Those belong to half of the measures shown in Table 3, while the rest ones are in Table 4. Based on the data in the tables, the obtained inconsistency values are various in the range of 0 and 7, from all test case measurements. The least value variant belongs to $\mathcal{I}_{\mathrm{d}}$ since it obtains one value only for all test cases. The most value variants belong to $\mathcal{I}_{\mathrm{mv}}$ and $\mathcal{I}\mathrm{D}_{\mathrm{MCS}}$ since they obtain 10 values for all test cases. Furthermore, there is no difference of values obtained by $\mathcal{I}_{\mathrm{mv}}$ and $\mathcal{I}\mathrm{D}_{\mathrm{MCS}}$.

3.3. Running Time

Comparisons of the required time to run the 36 OWL files with the use of three reasoners for each measure are here. The measures $\mathcal{I}_{\mathrm{d}}, \mathcal{I}_{\mathrm{D}_f}, \mathcal{I}_{\mathrm{mc}}$, and $\mathcal{I}\mathrm{D}_{\mathrm{MCS}}$ will represent this evaluation. There are sharp differences of the running time with each reasoner in the 9th, 13th, 16th, 18th, 19th, 22nd, 23rd, 27th, 28th, 29th, 30th, 36th of the test cases with the use of $\mathcal{I}_{\mathrm{d}}$, as depicted in Figure 5. The one in the 9th of the test cases tells that the required time to run the ontology with HermiT is 49 milliseconds, with JFact is 1 millisecond, with Pellet is 1 millisecond. The 18th of the test cases tells that the required time with Hermit is 53 milliseconds, with JFact is 2 milliseconds, and with Pellet is 2 milliseconds. HermiT reaches the highest running time among the reasoners in most of the test cases. Figure 6 also tells the same, in which HermiT reaches the highest running time among three reasoners in most of the test cases, with the use of $\mathcal{I}_{D_f}$. The peak of HermiT's running time is 1,803 milliseconds, belonging to the 25th of the test cases. Whereas it is 1,191 and 673 milliseconds for JFact and Pellet, respectively. In contrast, Pellet reaches the lowest running time in most of the test cases.

Table 3. Inconsistency Value Quantity of $\mathcal{I}_{\mathrm{d}}, \mathcal{I}_{\mathrm{MI}}, \mathcal{I}_{\mathrm{MI}^{\mathrm{C}}}, \mathcal{I}_{\mathrm{D}_f}$, And $\mathcal{I}_{\mathrm{p}}$ for the Test Cases

$\mathcal{I}_{\mathrm{d}}$		$\mathcal{I}_{\mathrm{MI}}$		$\mathcal{I}_{\mathrm{MI}^{\mathrm{C}}}$		$\mathcal{I}_{\mathrm{D}_f}$		$\mathcal{I}_{\mathrm{p}}$	
Inc. Value	Number of Ontologies	Inc. Value	Number of Ontologies	Inc. Value	Number of Ontologies	Inc. Value	Number of Ontologies	Inc. Value	Number of Ontologies
1	36	1	34	0.14	2	0	3	1	7
		2	2	0.17	3	0.01	5	2	10
				0.2	2	0.02	4	3	8
				0.25	4	0.17	2	4	4
				0.33	8	0.2	1	5	2
				0.5	8	0.25	1	6	3
				0.67	1	0.33	6	7	2
				1	6	0.5	6		
				2	2	1	8		

Table 4. Inconsistency Value Quantity of $\mathcal{I}_{\mathrm{IR}}, \mathcal{I}_{\mathrm{mc}}, \mathcal{I}_{\mathrm{nc}}, \mathcal{I}_{\mathrm{mv}}$, And $\mathcal{I}\mathrm{D}_{\mathrm{MCS}}$ for the Test Cases

$\mathcal{I}_{\mathrm{IR}}$		$\mathcal{I}_{\mathrm{mc}}$		$\mathcal{I}_{\mathrm{nc}}$		$\mathcal{I}_{\mathrm{mv}}$		$\mathcal{I}\mathrm{D}_{\mathrm{MCS}}$	
Inc. Value	Number of Ontologies	Inc. Value	Number of Ontologies	Inc. Value	Number of Ontologies	Inc. Value	Number of Ontologies	Inc. Value	Number of Ontologies
0.1	4	1	14	1	34	0.12	1	0.12	1
0.13	1	2	11	2	2	0.43	2	0.43	2

0.14	5	3	4	0.5	1	0.5	1
0.17	2	4	2	0.56	1	0.56	1
0.2	1	5	3	0.67	3	0.67	3
0.25	1	6	2	0.71	1	0.71	1
0.33	7			0.75	1	0.75	1
0.5	7			0.78	1	0.78	1
1	8			0.8	1	0.8	1
				1	24	1	24

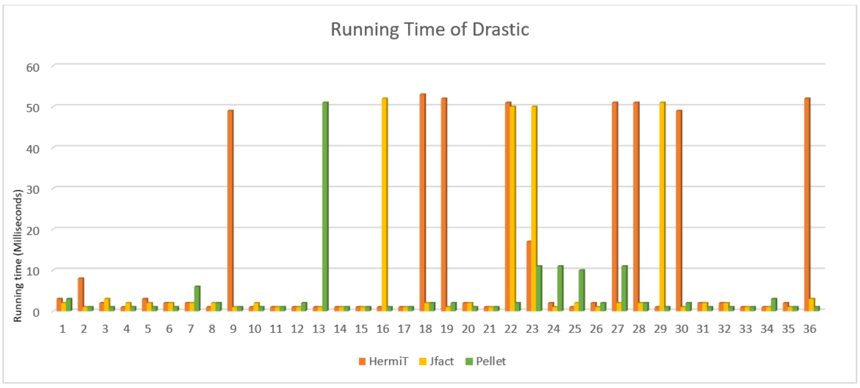


Figure 5. Running Time of $\mathcal{I}_{\mathrm{d}}$

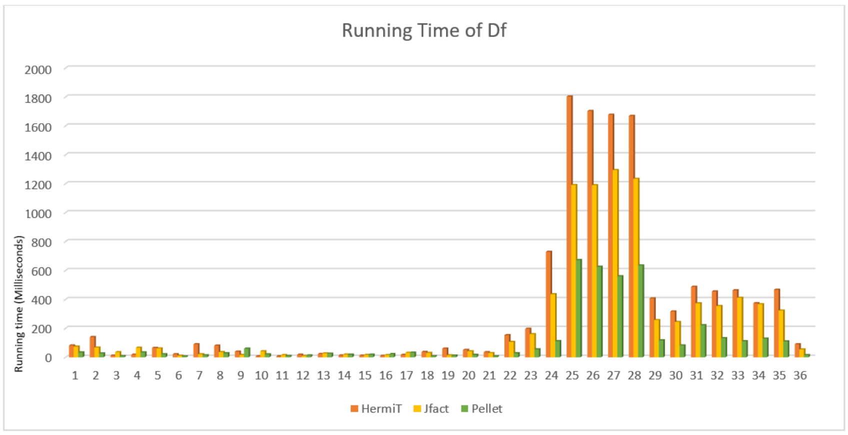


Figure 6. Running Time of $\mathcal{I}_{\mathrm{D}_f}$

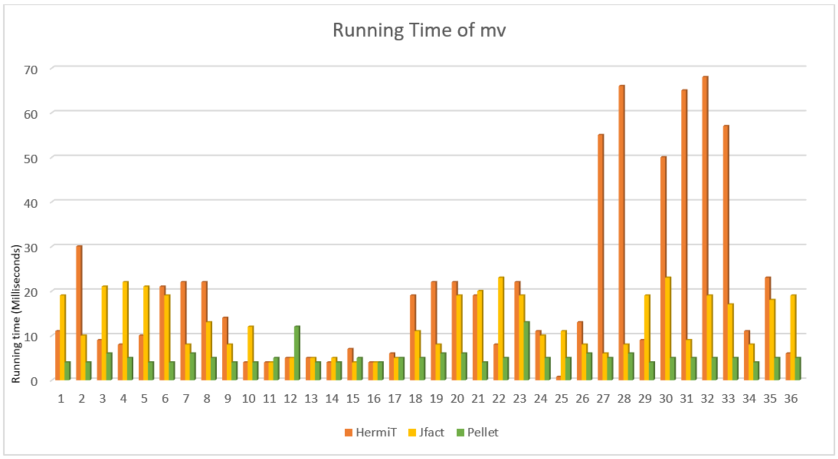


Figure 7. Running Time of $\mathcal{I}_{\mathrm{mv}}$

The one with the use of $\mathcal{I}_{\mathrm{mc}}$ shows no difference in general. That is HermiT takes the highest running time in most of the test cases as well, since it happens to more than 25 of 36 ontologies. Whereas JFact reaches the middle level of the running time and Pellet reaches the lowest running time in most of them. The same general case happens with the use of $\mathcal{I}_{\mathrm{mv}}$, even it is very sharp and clear to see, as depicted in Figure 7. The general claim also applies to the measurements with other measures in Onti Measures.

3.4. Size of Ontologies (wrt. the Running Time)

The evaluation of the size of ontologies wrt. the running time required by a measure with each of the reasoners will be depicted by a chart. For overall evaluation, there is a claim that the higher of the size will not impact the higher of the running time. Specifically, the higher of the size will not automatically affect to the higher of the running time with any reasoner. Since a chart for a measure confirms it, it is not necessary to provide 10 charts for the 10 measures.

The claim is clearly represented by Figure 8, specifically by some of its bars, e.g., the 11th and 25th of the test cases. The ontology size of the 11th of the test cases is 1, the running time with HermiT is 1 millisecond, the running time with JFact is 0 milliseconds (since it is rounded from some microseconds), and the running time with Pellet is 10 milliseconds. In contrast, the ontology size of the 25th is higher, but the running time with the two reasoners is not higher than the ones with the lower size. Particularly, the ontology size of the 25th is 10, the running time with HermiT is 1 millisecond, the running time with JFact is 2 milliseconds, and the running time with Pellet is 1 millisecond.

In the opposite, the condition happens to $\mathcal{I}_{\text{nc}}, \mathcal{I}\text{D}_{\text{MCS}}, \mathcal{I}_{\text{D}_f}, \mathcal{I}_{\text{p}}, \mathcal{I}_{\text{mc}}$ is different, that is the higher of the ontology size will impact the higher of the running time with the reasoners. Although this is the case, it cannot break the claim above since one case confirms it, then it is enough to approve it. Recall the claim above for overall condition in Onti Measures, it is obvious that the higher of the size does not impact to the higher of the running time with the three reasoners.

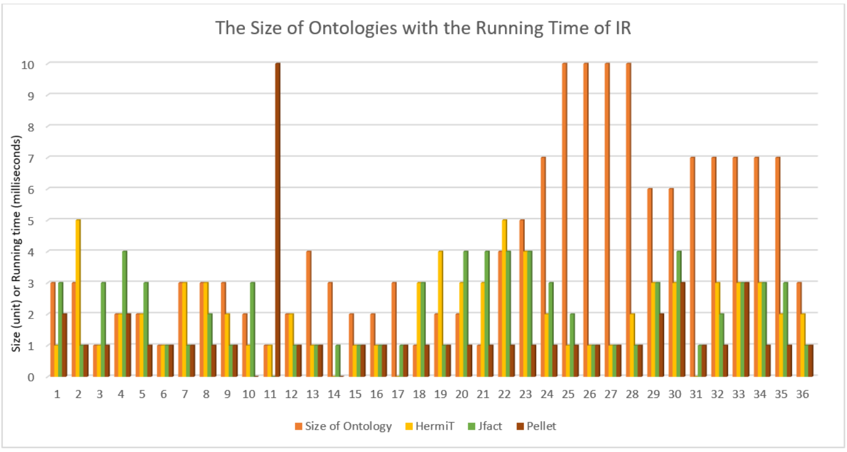


Figure 8. The Size of Ontologies with the Running Time of $\mathcal{I}_{\text{IR}}$

3.5. Descriptive Statistics

Table 5. Descriptive Statistics of Onti Measures for the Test Cases

Descriptive Analysis	Value of Descriptive Analysis									
	$\mathcal{I}_{\text{d}}$	$\mathcal{I}_{\text{MI}}$	$\mathcal{I}_{\text{MI}^\text{C}}$	$\mathcal{I}_{\text{D}_f}$	$\mathcal{I}_{\text{p}}$	$\mathcal{I}_{\text{IR}}$	$\mathcal{I}_{\text{mc}}$	$\mathcal{I}_{\text{nc}}$	$\mathcal{I}_{\text{mv}}$	$\mathcal{I}\text{D}_{\text{MCS}}$
Mean	1	1.08	0.52	0.37	3.14	0.42	2.31	1.06	0.86	0.86
Median	1	1	0.33	0.33	3	0.33	2	1	1	1
Mode	1	1	0.33	0.33	2	0.33	1	1	1	1
Standard Deviation	0	0.28	0.45	0.36	1.64	0.32	1.45	0.23	0.22	0.22
Range	0	1	1.86	1	6	0.9	5	1	0.88	0.88
Minimum	1	1	0.14	0	1	0.1	1	1	0.12	0.12
Maximum	1	2	2	1	7	1	6	2	1	1
Sum	36	39	18.86	13.23	113	15.17	83	38	31.09	31.09
Count	36	36	36	36	36	36	36	36	36	36

Descriptive statistics are discussed in this section to describe the basic features of the data collection from the test cases. There are 9 descriptive analyses in Table 5. They are mean, median, mode, standard deviation, range, maximum, minimum, sum, and count. Those analyses are chosen because they are commonly used. Mean is the average of the collected data. In this case, mean is the average of all inconsistency values. The smallest mean belongs to $\mathcal{I}_{\text{D}_f}$, whereas the biggest one belongs to $\mathcal{I}_{\text{p}}$. The median is the middle value in the set of the data, which is obtained by taking the value in the middle of the list after listing the data in numerical order. The smallest median belongs to $\mathcal{I}_{\text{MI}^\text{C}}, \mathcal{I}_{\text{D}_f}$, and $\mathcal{I}_{\text{IR}}$ and the biggest one belongs to $\mathcal{I}_{\text{p}}$.

Mode is the value which appears most often in the set of the values. The standard deviation is the amount of variation or dispersion of the set. Range is the difference between the smallest and the highest values in the set. Minimum is the smallest value, while maximum is the greatest or highest value. The smallest minimum belongs to $\mathcal{I}_{\text{D}_f}$ whereas the biggest ones belong to $\mathcal{I}_{\text{d}}, \mathcal{I}_{\text{MI}}, \mathcal{I}_{\text{p}}, \mathcal{I}_{\text{mc}}$, and $\mathcal{I}_{\text{nc}}$. Sum is the result of adding all values together, while count is the quantity of values in the set. There are five measures which have smallest maximum, i.e. $\mathcal{I}_{\mathrm{d}}, \mathcal{I}_{\mathrm{D}_f}, \mathcal{I}_{\mathrm{IR}}, \mathcal{I}_{\mathrm{mv}}$, and $\mathcal{I}\mathrm{D}_{\mathrm{MCS}}$, while the biggest one belongs to $\mathcal{I}_{\mathrm{p}}$..

4. DISCUSSIONS

The five kinds of evaluations have been done, i.e. inconsistency value (wrt. the reasoner), inconsistency value quantity, running time, size of ontologies (wrt. the running time), and descriptive statistics. They show that Pellet is better than the two other OWL reasoners, i.e. HermiT and Jfact. It is especially obtained from the evaluations of running time and the size of ontology wrt. the running time.

The comparison of the inconsistency measures based on the evalutions shows that $\mathcal{I}_{\mathrm{D_f}}$ is more efficient than other measures since it can give inconsistency degree in more details. In addition, it can produce precise limitation of minimum and maximum value, so that one can easier to get the certainty to judge which value is the worst one in any case. It is obtained from the evaluations of inconsistency value quantity and descriptive statistics.

This research can be compared to [13], [14] as the existing research that included the comparison about the 10 inconsistency measures related to HermiT, JFact, and Pellet. The difference among them is related to the test cases. The test cases belong to [13], [14] are ontologies for virus and disease cases, while the test cases belong to this research are inconsistent ontologies with no specific domain. The second difference among them is related to the parts to be evaluated. In [13], [14] the evaluations were done for the backend and frontend parts of Onti Measures, while in this research it is evaluated for the backend part only.

In [13], based on the running time, $\mathcal{I}_{\mathrm{mv}}$ and $\mathcal{I}\mathrm{D}_{\mathrm{MCS}}$ were the ones that took the longest time to run an ontology. The conclusion of reasoners performace did not exist. In [14], based on the performance of each reasoner in terms of processing ontology, the three reasoners have almost the same capabilities. Based on the resulting inconsistency value, Pellet is better than the other two reasoners. Based on the running time comparison, JFact is better than other reasoners. The measure $\mathcal{I}_{\mathrm{mv}}$ is the inconsistency measure in Onti Measures which requires the longest time for ontology inconsistency measurements. In addition, either in [13] or in [14], the higher of the size will not impact the higher of the running time. The latest claim is in line with the claim in this research.

The research [15], [16] compared the reasoners only. The research [16] showed that Pellet has the lowest response time. It is in line with the claim resulted by this research in which Pellet is better than other reasoners wrt. running time.

5. CONCLUSION

In this research, evaluations have been done for the Onti Measures that is an application program of ontology-based inconsistency measures for OWL ontologies. Onti Measures contains 10 inconsistency measures. The evaluations are done for all of them with the use of HermiT, JFact, and Pellet as the OWL reasoners and 36 inconsistent OWL 2 ontologies as the test cases.

Evaluation of inconsistency value wrt. the reasoner claims that the same inconsistent OWL file that is computed with different measure in Onti Measures may result in the different inconsistency value. Any chosen OWL reasoner does not impact the difference. In terms of evaluation of inconsistency value quantity, the least value variant belongs to $\mathcal{I}_{\mathrm{d}}$ and the most value variants belong to $\mathcal{I}_{mv}$ and $\mathcal{I}\mathrm{D}_{\mathrm{MCS}}$. Evaluation of running time claims that Onti Measures with HermiT takes the longest time to run in most of the cases. In terms of evaluation of the size of the ontologies wrt. the running time, the higher of the size does not impact to the higher of the running time with any reasoner The evaluation of the descriptive statistics has shown analysis result values for 9 analyses, i.e., mean, median, mode, standard deviation, range, minimum, maximum, sum, and count. Overall, the evaluations show that Pellet is better than the two other OWL reasoners and $\mathcal{I}_{\mathrm{D_f}}$ is more efficient than the other measures.

REFERENCES

[1] M. Krötzsch, F. Simančík, and I. Horrocks, "A description logic primer," in *Perspectives on Ontology Learning*, vol. 18, 2014.

[2] A. Hunter and S. Konieczny, "Approaches to measuring inconsistent information," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2004. doi: 10.1007/978-3-540-30597-2_7.

[3] A. Hunter and S. Konieczny, "Shapley inconsistency values," in *Proceedings of the International Conference on Knowledge Representation and Reasoning*, 2006.

[4] A. Hunter and S. Konieczny, "Measuring inconsistency through minimal inconsistent sets," in *Proceedings of the International Conference on Knowledge Representation and Reasoning*, 2008.

[5] M. Thimm, "On the expressivity of inconsistency measures," *Artif Intell*, vol. 234, 2016, doi: 10.1016/j.artint.2016.01.013.

[6] Y. Ma, G. Qi, and P. Hitzler, "Computing inconsistency measure based on paraconsistent semantics," *Journal of Logic and Computation*, vol. 21, no. 6, 2011, doi: 10.1093/logcom/exq053.

[7] J. Grant and A. Hunter, "Analysing inconsistent first-order knowledgebases," *Artif Intell*, vol. 172, no. 8–9, 2008, doi: 10.1016/j.artint.2007.11.006.

[8] N. A. Ekowati, "Inconsistency Measures for OWL Ontologies," Technische Universität Dresden, Dresden, 2017.

[9] K. Mu, W. Liu, Z. Jin, and D. Bell, "A Syntax-based approach to measuring the degree of inconsistency for belief bases," *International Journal of Approximate Reasoning*, vol. 52, no. 7, 2011, doi: 10.1016/j.ijar.2011.04.001.

[10] J. Grant and A. Hunter, "Measuring consistency gain and information loss in stepwise inconsistency resolution," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2011. doi: 10.1007/978-3-642-22152-1_31.

[11] D. Doder, M. Raković, Z. Marković, and Z. Ognjanović, "Measures of inconsistency and defaults," *International Journal of Approximate Reasoning*, vol. 51, no. 7, 2010, doi: 10.1016/j.ijar.2010.05.007.

[12] G. Xiao and Y. Ma, "Inconsistency measurement based on variables in minimal unsatisfiable subsets," in *Frontiers in Artificial Intelligence and Applications*, 2012. doi: 10.3233/978-1-61499-098-7-864.

[13] Nur Alfi Ekowati, Ika Indah Lestari, and Sulistiyasni, "Pengembangan Onti Measures Berbasis Web dengan Pengujian Data Ontology Virus dan Penyakit," *Jurnal Nasional Teknik Elektro dan Teknologi Informasi*, vol. 10, no. 4, 2021, doi: 10.22146/jnteti.v10i4.2443.

[14] Ika Indah Lestari, Nur Alfi Ekowati, and S. Sulistiyasni, "Descriptive Analysis and Comparison of Reasoner Using Onti Measures," *Jurnal Teknik Informatika (Jutif)*, vol. 5, no. 1, pp. 301–312, Feb. 2024, doi: 10.52436/1.jutif.2024.5.1.1839.

[15] S. Abburu, "A Survey on Ontology Reasoners and Comparison," *Int J Comput Appl*, vol. 57, no. 17, 2012.

[16] A. Khamparia and B. Pandey, "Comprehensive analysis of semantic web reasoners and tools: a survey," *Educ Inf Technol (Dordr)*, vol. 22, no. 6, 2017, doi: 10.1007/s10639-017-9574-5.

[17] H. Liu and H. B. Kuan Tan, "Covering code behavior on input validation in functional testing," *Inf Softw Technol*, vol. 51, no. 2, 2009, doi: 10.1016/j.infsof.2008.07.001.

[18] S. Nidhra, "Black Box and White Box Testing Techniques - A Literature Review," *International Journal of Embedded Systems and Applications*, vol. 2, no. 2, pp. 29–50, Jun. 2012, doi: 10.5121/ijesa.2012.2204.

[19] "OWL Test Cases." Accessed: May 22, 2023. [Online]. Available: http://owl.semanticweb.org

[20] S. Rudolph, "Foundations of description logics," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2011. doi: 10.1007/978-3-642-23032-5_2.