# Evaluation of Ontologies and DL Reasoners

**Muhammad Fahad, Muhammad Abdul Qadir and Syed Adnan Hussain Shah**

Mohammad Ali Jinnah University, Islamabad, Pakistan.

mhd.fahad@gmail.com, {aqadir,adnan}@jinnah.edu.pk

**Abstract:** Ontology driven architecture has revolutionized the inference system by allowing interoperability and efficient reasoning between heterogeneous multi-vendors systems. Sound reasoning support is highly important for sound semantic web ontologies which can only be possible if state-of-the-art Description Logic Reasoners were capable enough to identify inconsistency and classify taxonomy in ontologies. We have discussed existing ontological errors and design anomalies, and provided a case study incorporating these errors. We have evaluated consistency, subsumption, and satisfiability of DL reasoners on the case study. Experiment with DL reasoners opens up number of issues that were not incorporated within their followed algorithms. Especially circulatory errors and various types of semantic inconsistency errors that may cause serious side effects need to be detected by DL reasoners for sound reasoning from ontologies. The evaluation of DL reasoners on Automobile ontology helps in updating the subsumption, satisfiability and consistency checking algorithms for OWL ontologies, especially the new constructs of OWL 1.1.

## 1 Introduction

Ontology has revolutionized the inference system by allowing interoperability between heterogeneous multi-vendors systems and semantic web applications [17]. Well formed ontologies can only furnish the semantics for emerging semantic web and provide reasoning capability that they require. Due to their expressive power and reasoning capabilities, they are being used in wide range of applications and knowledge based systems [1]. Like any other dependable component of a system, Ontology has to go through a repetitive process of refinement and evaluation during its development lifecycle so that they can serve their purposes and make their user safer in the application.

Several approaches for evaluation of taxonomic knowledge on ontologies are contributed in the research literature. Ontologies can be evaluated by considering design principles [6,7], maintenance issues [2] , use in an application [13] and predictions from their results, peer review [16] , comparison with a golden stan-

dard [10] or reference ontology library [13] or manipulation of data [3]. These approaches evaluate the ontologies from different frame of references that enable better reasoning support for fulfillment of sound semantic web vision.

Sound semantic web ontologies have to create balance between computational complexity needed for reasoning mechanisms and expressive power of the concepts defined [11]. Initial ontologies RDF and RDFS [1], provide very limited expressive power that is not very rich to represent semantics related with a domain. OWL and its newly formed specie OWL 1.1 provide much expressive power and sparked the inference engines by providing the suitable reasoning support. Though the DL reasoners were long before the existence of OWL ontologies, but new dialects of OWL ontologies need some more reasoning and inference services. These languages especially OWL 1.1 opens up new challenges [21] for DL reasoners to check subsumption, consistency and satisfiability from ontologies developed for sound semantic web environment.

One of the benchmark for DL reasoners is presented by Pan with realistic ontologies to check the time taken by these reasoners [20]. The experiment was dealt with 135 ontologies, and DL reasoners timeout and aborted operations were counted and reported. The experiment helped out in optimizing the algorithms followed by DL reasoners. The one conducted by us differs and is unique as it helps out identifying deficiencies and incompleteness in their algorithms.

This paper is based on our line of research [4,5,12,14,15] on evaluation of ontologies. In [17, 18], we presented the ontology evaluation framework for OWL ontologies and extended the ontology error taxonomy initially formed by Gomez. In this paper, we provide a case study on design anomalies and taxonomic errors. We have formed Automobile ontology and seeded all types of errors to promote learning and understandability of ontological errors. This ontology also acts as a test data for evaluation of Description Logic Reasoners, and helps in finding some of the deficiencies in the algorithms followed by them.

Rest of the paper is organized as follows: section 2 presents the types of ontological error and their classification. The same section builds the case study of Automobile ontology with these errors. Section 3 discusses the evaluation of state-of-the-art DL reasoners and our experiment details. Section 4 concludes the paper.

## 2 Taxonomic Errors and Design Anomalies

Gomez-Perez [6,7,8] identified three main classes of taxonomic errors that might occur when modelling the conceptualization into taxonomies. These classes of errors are Inconsistency, Incompleteness and Redundancy. We have extended these classes by incorporating more errors in each class by evaluation of online ontologies [17,18]. We seeded these errors in automobile ontology, as shown in Fig.1, which acts as benchmark for DL reasoners later on. Table 1 and 2 provide

the important axioms for understanding the ontological errors. The top level description of errors in automobile ontology is provided by subsections.

Table 1. Important axioms of concepts in Automobile ontology.

| |
|---|
| Owner ⊆ ∃ drives MotorVehicle |
| Passenger ⊆ ∃ involves MotorVehicle ∏ ∃ hasreserved MotorVehicle |
| VehicleOwner ⊆ owns ≥ 1 MotorVehicle |
| NoOwner ⊆ owns = 0 MotorVehicle |
| Owner2Vehicle ⊆ owns = 2 PassengerVehicle |
| Owner4Vehicle ⊆ owns = 4 PassengerVehicle |
| OwnerManyVehicle ⊆ owns ≥ 3 PessangerVehicle ∏ owns≤ 8 PassengerVehicle |
| Ownerlessthan3Vehicle ⊆ owns ≤ 2 PassengerVehicle |
| OwnerSomeVehicle ⊆ ∃ owns PassengerVehicle |
| OnwerAllVehicle ⊆ ∀ owns PassengerVehicle |
| BikeOwner ⊆ ∃ hasBike Bike |
| Bike ⊆ HondaMotorBike U YamahaMotorBike |
| Men ⊆ Peson ∏ ∀ hasGender Male |
| Women ⊆ Peson ∏ ∀ hasGender Female |

Table 2. Disjointness and Property information in Automobile ontology.

| **Property(Domain, Range)** | **Disjoint Axioms (Class1 ┤ Class2)** |
|---|---|
| owns(VehicleOwner, MotorVehicle) | Driver ┤ Passenger |
| owns(Owner, MotorVehcicle) | MotorVehicle ┤ Plane |
| drives(Driver, MotorVehicle) | PIA ┤ Truck |
| hasReserved(Driver, MotorVehicle) | Male ┤ Female |
| involves(Pessenger, MotorVehicle) | Coach ┤ Van |
| Functional hasBike(BikeOnwer, Bike) | Pejjero ┤ Jeep |
| hasRegisterationNo(MotorVehicle, RegisterationNo) | YamahaMotorBike ┤ HondaMotorBike |

### 2.1 Inconsistency Errors

There are mainly three types of errors that cause inconsistency and contradictions during the reasoning from the ontology. These are Circulatory errors, Partition errors and Semantic inconsistency errors.

Circulatory errors occur when a class is defined as a subclass or superclass of itself at any level of hierarchy in the ontology [7]. In automobile ontology, circulatory error on concept *Class_2* occurs as it is specified as subclass of *Class_5*. OWL ontologies provide constructs to form property hierarchies by specifying *MobilinkNo* as subproperty of *MobileNo* and *MobileNo* as subproperty of *ContactNo*. Circulatory error in property hierarchy [17] occurs by specifying *MobilinkNo* as subproperty of *ContactNo*.

Partition errors occur while decomposition of concept into many subconcept. A common class/property/instance in disjoint decomposition and partition error occurs when ontologists create the class/instance (or property) that belongs to various disjoint classes (or disjoint properties) [7]. In automobile ontology, *MiniVan* as subclass of two disjoint classes *Coach* and *Van* creates inconsistency of this type. Likewise common property in disjoint decomposition of properties creates inconsistency in property hierarchy [17]. We seeded instance *myMiniVan123* which serves as common instance between disjoint classes. Moreover when concepts are disjoint then they should not use the properties of their disjoint concepts. Concept *Passenger* ($\subseteq \exists$ *hasReserved MotorVehicle*) being disjoint with *Driver* constitutes this type of inconsistency by using property *hasReserved*.

External instance in exhaustive decomposition occurs when one or more instances of base class do not belong to any of the subclasses [7]. In automobile ontology, we seeded *SaudiAirWays* as instance of *Plane* that does not belong to *PIA* and *BritishAirWays* subclasses.

Semantic Inconsistency errors occur when ontologists make an incorrect class hierarchy by classifying a concept as a subclass of a concept to which it does not really belong [7]. For example the ontologist classifies the concept *Airbus* as a subclass of the concept *Train*. Or the same might have happened when classifying instances. We identify mainly three reasons due to which incorrect semantic classification originates [5] and categorized Semantic inconsistency errors into three subclasses.

Weaker domain specified by subclass error [5] occurs when classes that represent the larger domain are kept subclasses of concept that possess smaller domain. In automobile ontology, the semantic inconsistency of this type occurs as more generalized concept $OnwerSomeVehicle \subseteq \exists$ *owns PassengerVehicle* is created as a subclass of the concept *Onwer4vehicle* $\subseteq$ *owns = 4 PassengerVehicle*.

Domain breach specified by subclass error [5] occurs when a subclass adds more features but the additional features are violating the existing features of their superclasses. In automobile ontology, *OwnerManyVehicle* $\subseteq$ *owns* $\geq$ *3 PessangerVehicle* $\prod \leq$ *8 PassengerVehicle* concept as a subclass of *Onwer4Vehicle* concept breaches the domain.

Disjoint domain specified by subclass error [5] occur when ontologists specify concept as subclass of a concept having disjoint domain. In automobile ontology, *NoOwner* concept as a subclass of *OnwerVehicle* concept, *Onwer2Vechicle* and *Onwerlessthan3Vehicle* as subclasses of *Onwer4Vehicle* shows the disjoint domain specified by subclass error. Moreover, *Women* $\subseteq$ *Person* $\prod \forall$ *hasGender.Female* as a subclass of *Men* $\subseteq$ *Person* $\prod \forall$ *hasGender.Male*, as *Male* is disjoint with *Female* constitutes the error of this category. Similarly, these semantic inconsistency errors can be applied same to the instances of superclasses and subclasses to check whether they have conformance with each other.

### 2.2 Incompleteness Errors

Sometimes ontologists classify concepts but overlook some of the important information about them. Such incompleteness often creates ambiguity and lacks reasoning mechanisms. The following subsections give the overview of incompleteness errors.

Incomplete Concept Classification error [7] occurs when ontologists overlook some of the concepts present in the domain while classification of particular concept. In automobile ontology, *Plane* concept is incompletely classified by ignoring *SaudiAirways*, *ShaheenExpress, etc,* types of planes.

Partition Errors occur when ontologist omits important axioms or information about the classification of concept. Disjoint Knowledge Omission error [7] occurs when ontologists classify the concept into many subclasses, but omits disjoint knowledge axiom between them. In automobile ontology, disjoint axiom between *PassengerVehicle* and *LoaderVehicle* is ignored. We experienced catastrophic results by disjoint knowledge omission between user and Administrator in *Access_Policy* ontology [14]. Similarly disjoint axiom between properties create incompleteness error in property partitioning [17].

Exhaustive knowledge Omission occurs when ontologists do not follow the completeness constraint while decomposition of concept into subclasses [7]. In automobile ontology, ontologist models the *Coach*, and *Van* classes as disjoint subclasses of *PassengerVehicle* concept, but does not specify that whether this classification forms an exhaustive decomposition.

For powerful reasoning and enhanced inference, OWL ontology provides some tags that can be associated with properties of classes [1]. OWL functional and inverse-functional tags associated with properties indicate how many times a domain concept can be associated with range concept via a property. Sometimes ontologists do not give significance to these property tags and do not declare datatype or object properties as functional or inverse-functional. As a result machine can not reason about a property effectively leading to serious complications [15]. In automobile ontology, *hasRegisterationNo* as an object property between *MotorVehicle* and *RegisterationNo* is an example of functional object property due to the fact that every subject *Vehicle* has only one registeration number. Ignoring Functional tag with *RegisterationNo* allows property to have more than one values leading to inconsistency. One of the main reason for such inconsistency is that ontologist has ignored that OWL ontology by default supports multi-values for datatype property and object property. In this example, *hasRegisterationNo* property also needs to be specified as inverse-functional property as it uniquely identifies the subject.

Sufficient knowledge Omission Error (SKO) [12] occurs when concept has only *Necessary description*, i.e., defined only by the basic criteria of subclass-of, and does not have *sufficient description* that elaborates the characteristics of concept and defines the context in terms of other concepts. In automobile ontology, *PIA* and *BritishAirWays* concepts need sufficient knowledge to interpret and distinguish them while reasoning.

### 2.3 Redundancy Errors

Redundancy occurs when particular information is inferred more than once from the relations, classes and instances found in ontology. The following are the types of redundancies that might be made when developing taxonomies.
Redundancies of *SubclassOf* error [7] occur when ontologists specify classes that have more than one *SubclassOf* relation directly or indirectly. In automobile ontology specifying *Jeep* as a subclass of *PassengerVehicle* and *MotorVehicle*, creates redundancy as *PassengerVehicle* is already a *subclassOf MotorVechicle*. Here indirect *SubclassOf* relation exists between *Jeep* and *MotorVehicle* creating redundancy of this type. Similarly, redundancy of *SubpropertyOf* can exist while building property hierarchies [17]. Redundancies of *InstanceOf* relation [7] occur when ontologists specify instance *myJeep* as an *InstanceOf MotorVehicle* and *PassengerVehicle* concepts.

Identical formal definition of classes, properties or instances may occur when ontologist defines different (or same) names of two classes, properties or instances respectively, but provides the same formal definition. In automobile ontology, $Driver \subseteq \exists\ drives\ MotorVehicle$ and $Owner \subseteq \exists\ drives\ MotorVehicle$ specifies identical formal definition of classes.

Redundancy of Disjoint Relation (RDR) [12] occurs when the concept is explicitly defined as disjoint with other concepts more than once. In automobile ontology, disjoint axiom between *PIA* and *Truck* creates RDR error as *MotorVehicle* and *Plane* concepts were already disjoint with each other. Fig. 1 shows the class hierarchy of Automobile ontology.

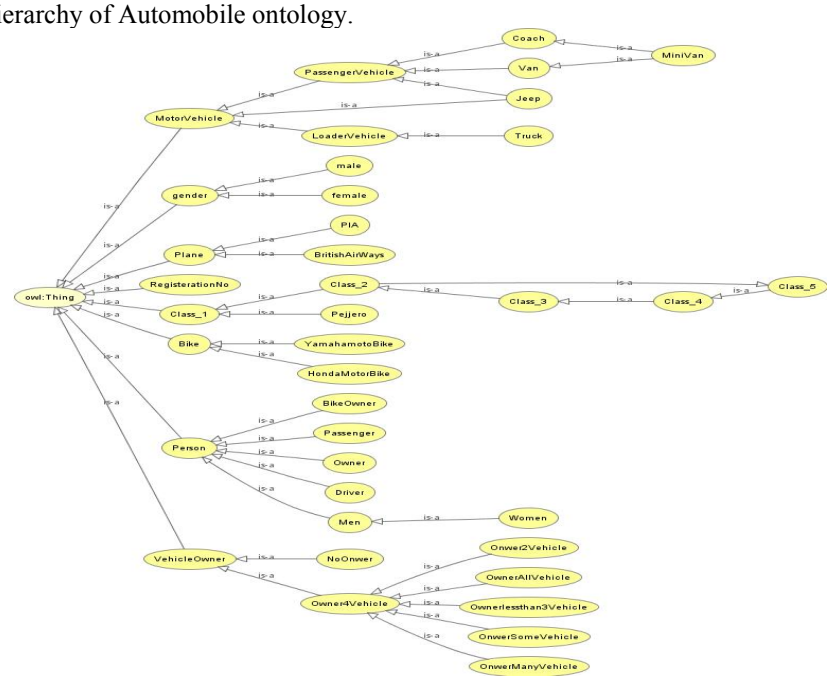


**Fig. 1. Class hierarchy of Automobile ontology.**

### 2.4 Design Anomalies in Ontologies

Besides taxonomic errors, Baumeister and Seipel [2] identified some design anomalies that prohibit simplicity and maintainability of taxonomic structures within ontology. These do not cause inaccurate reasoning about concepts, but point to problematic and badly designed areas in ontology. Identification and removal of these anomalies should be necessary for improving the usability, and providing better maintainability of ontology.

**Property Clumps:** The repeated group of datatype properties (name, model, color, price, etc.) in class *MotorVehicle* and *Bike* create property clump.

**Chain of Inheritance:** In automobile ontology, *Class_1* to *Class_5* creates chain of inheritance as these concepts have no appropriate descriptions in the ontology except inherited child.

**Lazy Concepts:** A leaf concept that is not instantiated and never used in the application is called the lazy concept. In automobile ontology we have created many lazy concepts, *Truck* concept is one of the example of this kind.

**Lonely Disjoints:** We moved the concept *Pejjero* from the *PassengerVehicle* hierarchy somewhere in the other hierarchy, creating lonely disjoint with *Jeep* concept.

## 3 Description Logic Reasoners and Ontology Errors

In this section, we are evaluating the state-of-the-art Description Logic (DL) reasoners by providing them the ontology seeded with the errors described in above section 2. The evaluation of DL reasoners helps us to build more powerful algorithms so that reasoning from ontologies can be enhanced to fulfill the goals of semantic web. The experiment was performed on three state-of-the-art DL reasoners Pellet, FaCT++, and Racer. The salient features of these reasoners and experiment details are explained below.

**Racer:** Racer was implemented in Lisp to demonstrate the tableaux calculus for SHIQ, and follows the multiple optimization strategies for better reasoning support including dependency-directed backtracking, transformation of axioms, model caching and merging, etc, [19].

**Pellet:** Pellet employs reasoning on SHIN (D) and SHON (D) and implemented in Java with the strategies of TBox partitioning, nominal support, absorption, semantic branching, lazy unfolding, dependency directed backjumping [21]. Datatype reasoning, individual reasoning, and optimization in Abox query answering makes it more attractive for sound semantic web applications.

**FaCT++:** FaCT++ [22] an improved version of FaCT [23] employs tableaux algorithms for SHOIQ description logic and implemented in C++ but has very limited user interface and services as compared to other reasoners. The strategies followed are absorption, model merging, told cycle elimination, synonym replacement, ordering heuristics and taxonomic classification.

**Experiment Discussion:** The automobile ontology that was seeded with various types of errors is taken as the test data. These errors and anomalies were seeded very intelligently so that performance of consistency, subsumption, satisfiability, and tableaux algorithm efficiency can be measured. Due to space limitations, the only necessary errors were discussed above and here also we discuss our high level findings from experiment. The experiment was conducted on FaCT++1.1.11 (uploaded date: March 28, 08), Pellet 1.5.1 (uploaded date: Oct 26, 07) and Racer 1.9.0 versions.

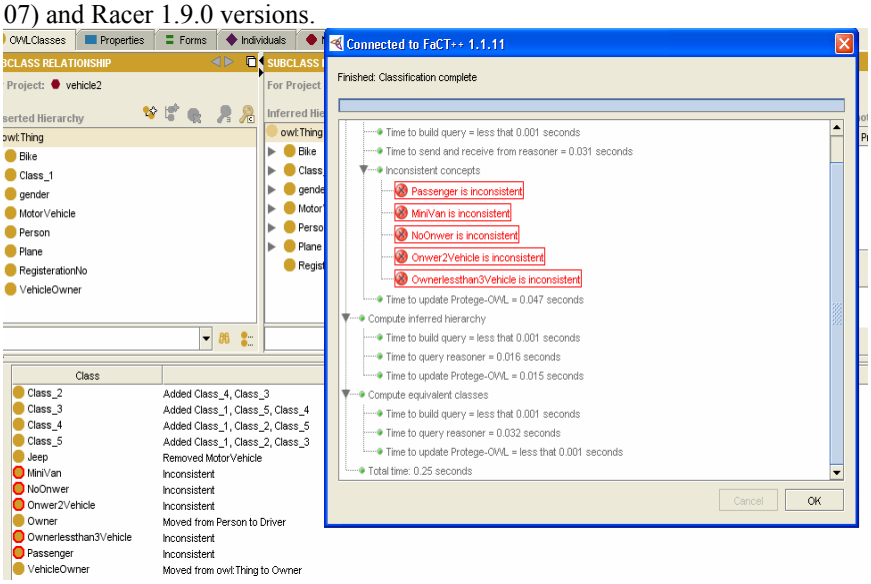


**Fig. 2. Results produced by FaCT++ Reasoner**

By consistency checking on automobile ontology, only some of the errors were detected. The inconsistent concepts (*Passenger, MiniVehicle, Owner2Vehicle, NoOwner, Onwerlessthan3Vehicle*) were detected by all the three DL reasoners. These inconsistent concepts are described along the errors above. Inconsistency of type common property in disjoint decomposition of properties is detected by only FaCT++. Redundancy of subclassOf error on concept *Jeep* was detected and in classified taxonomy superclass *MotorVehicle* was removed, as shown in classify taxonomy results *'Removed Motorvehicle'* in left down side of Figure 2. But some very important situations were not detected highlighting their deficiencies. One of the important aspects during reasoning from ontologies is that it should detect circles from taxonomies and get himself out from traversing circles again and again. According to Gomez [7], circle in hierarchy is error and should be detected and removed. This experiment highlights deficiency in their algorithms that they are not capable of handling circulatory errors in class hierarchies and property hierarchies. Circulatory error at *Class_2* was detected and all the subsumptions were reasoned as shown in the inferred class hierarchy by FaCT++ DL reasoners in

Figure 3, and added superclasses for all the classes in the circle as shown in Figure 2 (down left side). We again performed this experiment by making a circulatory error of distance 10. This time the inferred class hierarchy looks like a network of subsumption relations, making infeasible reasoning. Again performing the same experiment with a circulatory error with long chain of inheritance made the three DL reasoners crash. Imagine the consequences of circulatory error in ontology developed for critical application where strong reasoning with shorter answer time is required.

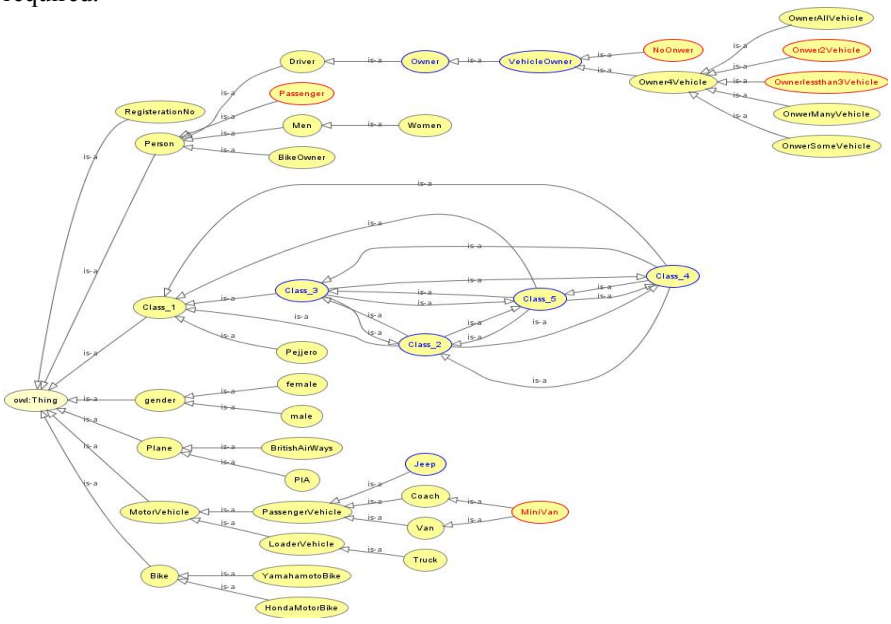


Fig. 3. Inferred class hierarchy

Besides circulatory, these reasoners have not identified various types of semantic inconsistency errors. Especially concept *Women* as a subclass of *Men*, with explicit description about disjointness between *Male* and *Female* can be detected as they also specify disjoint domains, but was not detected. Domain breach by specifying *OnwerManyVehicle* as subclass of *Onwer4Vehicle*, and *OnwerSomeVehicle* as weaker domain specified by subclass errors were also not detected.

Incompleteness as informal errors were not detected, as we know that incompleteness can only be manually detected by tester's domain knowledge and analysis of spot spots of ontologies by previous knowledge or populated data. Functional Property omission with *hasRegisterationNo* allowed creating several inconsistent registration numbers for a single vehicle. Disjoint knowledge omission, sufficient knowledge omission and exhaustive knowledge omission errors were also not detected.

Although the redundancy of subclass-of is detected and the arc from concept *Jeep* to concept *MotorVehicle* was deleted in the inferred hierarchy, but this was not always desirable. Sometimes that arc (between concept c and its ancestor) would be the actual one and the other arc (between concept c and its parent) would be the erroneous, but DL reasoners always does the same. Other types of redundancy like redundancy of disjoint relations, identical formal definitions were also not detected. On basis of common property *owns*, concept *VehicleOwner* is inferred as subclass of *Owner* concept and the hierarchy of *VehicleOwner* is moved to it. The overall experiment concludes that current state-of-the-art DL reasoners should be upgraded with respect of these errors and anomalies. Sound reasoning support is highly important for sound semantic web environment which can only be possible if these reasoners were capable enough identifying inconsistencies in ontologies.

## 5 Conclusion

Ontology driven architecture has revolutionized the inference system by allowing interoperability between heterogeneous multi-vendors systems. We have identified that accurate ontologies free from errors enable more intelligent interoperability, provide better reasoning mechanisms, improve the accuracy of ontology mapping and merging and combined use of them can be made possible. We have discussed existing ontological errors and design anomalies, and provided a case study that promotes understanding about these. Experiment with DL reasoners opens up number of issues that were not incorporated within their followed algorithms. Especially circulatory errors and various types of semantic inconsistency errors may cause serious side effects, and need to be detected by DL reasoners for sound reasoning from ontologies. The evaluation of DL reasoners on Automobile ontology helps in updating the subsumption, satisfiability and consistency checking algorithms for OWL ontologies, especially the new constructs of OWL 1.1.

## References

1. G. Antoniou, and F.V. Harmelen, A Semantic Web Primer. MIT Press Cambridge, ISBN 0-262-01210-3, 2004.
2. J. Baumeister, and D.S. Seipel, Owls–Design Anomalies in Ontologies, 18th Intl. Floida Artificial Intelligence Research Society Conference (FLAIRS), pp 251-220, 2005.
3. C. Brewster et al, Data driven ontology evaluation. Proceedings of Intl. Conf. on Language Resources and Evaluation, Lisbon, 2004.
4. M. Fahad, M.A. Qadir, M.W. Noshairwan, N. Iftikhar,. DKP-OM: A Semantic Based Ontology Merger. In Proc. 3rd International conference on Semantic Technologies, I-Semantics 5-7 September 2007, Journal of Universal Computer Science (J.UCS). 2007a

5. M. Fahad, M.A. Qadir, W. Noshairwan, Semantic Inconsistency Errors in Ontologies. Proc. of GRC 07, Silicon Valley USA. IEEE CS. pp 283-286, 2007b.
6. A. Gomez-Perez, Some ideas and examples to evaluate ontologies. KSL, Stanford University., 1994.
7. A. Gomez-Perez, M.F. Lopez, and O.C. Garcia, Ontological Engineering: With Examples from the Areas of Knowledge Management, E-Commerce and the Semantic Web. Springer ISBN:1-85253-55j-3, 2001.
8. A. Gomez-Perez et al., Evaluation of Taxonomic Knowledge on Ontologies and Knowledge-Based Systems. Intl. Workshop on Knowledge Acquisition, Modeling and Management., 1999.
9. C. Jelmini, and S. M-Maillet, OWL-based reasoning with retractable inference", In RIAO Conference Proceedings 2004.
10. A. Maedche and S. Staab, Measuring similarity betwe- en ontologies. Proc. CIKM 2002. LNAI vol. 2473, 2002.
11. D. Nardi, et al. 2000. The Description Logic Handbook: Theory, Implementation, and Applications.
12. W. Noshairwan, M.A. Qadir, M.A., M. Fahad, Sufficient Knowledge Omission error and Redundant Disjoint Relation in Ontology. InProc. 5th Atlantic Web Intelligence Conference June 25-27, France, 2007a.
13. R. Porzel, R. Malaka, A task-based approach for ontology evaluation. ECAI Workshop Ont. Learning and Population, 2004.
14. M.A. Qadir, W. Noshairwan, Warnings for Disjoint Knowledge Omission in Ontologies. Second International Conference on internet and Web Applications and Services (ICIW07). IEEE, p. 45, 2007a.
15. M.A. Qadir, M. Fahad, S.A.H. Shah, Incompleteness Errors in Ontologies. Proc. of Intl GRC 07, USA. IEEE Computer Society. pp 279-282, 2007b.
16. K. Supekar, A peer-review approach for ontology evaluation. Proc. 8th Intl. Protégé Conference, Madrid, Spain, July 18–21, 2005.
17. M. Fahad, and M.A. Qadir, A Framework for ontology evaluation. 16th Intl. Proceeding of Conceptual Structures. July 2008, France. Vol-354, pages 149-158, 2008a.
18. M. Fahad, M.A. Qadir, M.W. Noshairwan, Ontological Errors: Inconsistency, Incompleteness and Redundancy. (to appear) In proc. 10th International Conference on Enterprise Information Systems (ICEIS'08). June 2008. Barcelona, Spain, 2008b.
19. V. Haarslev, R. M¨oller, Racer system description. In Gor´e, R., Leitsch, A., Nipkow, T., eds.: International Joint Conference on Automated Reasoning, IJCAR' 2001, June 18-23, Siena, Italy, Springer-Verlag (2001) 701–705
20. Z. Pan, Benchmarking DL Reasoners Using Realistic Ontologies. Bell Labs Research and Lehigh University, 2007
21. B. Parsia, E. Sirin, Pellet: An owl dl reasoner. In: Proc. International Semantic Web Conference. (2005)
22. I. Horrocks, U. Sattler, A tableaux decision procedure for SHOIQ. In: Proceedings of Nineteenth International Joint Conference on Artificial Intelligence. (2005)
23. I. Horrocks, The FaCT System. International conference. on Analytic Tableaux and Related Methods (TABLEAUX'98), pp 307-312,vol 1397, Springer-Verlag, 1998