Towards a benchmark for Semantic Web reasoners - an analysis of the DAML ontology library

Christoph Tempich and Raphael Volz

Institute AIFB, University of Karlsruhe, Germany
http://www.aifb.uni-karlsruhe.de
(tempich,volz)@aifb.uni-karlsruhe.de

1 Introduction

Benchmarks are one important aspect of performance evaluation. This paper concentrates on the development of a representative benchmark for Semantic Web-type[1] ontologies. To this extent we perform a statistical analysis of available Semantic Web ontologies, in our case the DAML ontology library, and derive parameters that can be used for the generation of synthetic ontologies. These synthetic ontologies can be used as workloads in benchmarks.

Naturally, performance evaluation can also be performed using a real workload, viz. a workload that is observed on a reasoner being used for normal operations. However, such workloads can usually not be applied repeatedly in a controlled manner.

Therefore synthetic workloads are typically used in performance evaluations. Synthetic workloads should be a representation or model of the real workload. Hence, it is necessary to measure and characterize the workload on existing reasoners to produce meaningful synthetic workloads.

This should allow us to systematically evaluate different reasoners and reasoning techniques using a benchmark to gain realistic practical comparisons of individual systems.

1.1 Related Work

The development of benchmarks for ontology-based systems is substantially different from the development of a test suite [3] for testing the correctness or ability of a reasoner in handling particular primitives of an ontology language. The latter is intended to give yes or no answers to questions like whether a system can make certain entailments or find particular inconsistencies. The former, however, is intended to come up with numbers for a set of performance criteria (metrics).

Within the Description Logic community benchmarking [8, 6] was performed repeatedly in the past for empirical system comparsion. However, these representativeness of the used benchmarkss [1, 7, 6] are questionable for practical cases

[1] Hence RDFS, DAML+OIL and OWL

due to several reasons. For example, [8] tested the performance of class satisfiability based on a sequence of classes which are (exponentially) increasingly difficult to compute. These class definitions are hardly representative for practical cases. The test for ABox reasoning was underdeveloped since most systems at the time of evaluation did not support any ABox reasoning capabilities.

[6] used both real and synthetically generated knowledge bases as one part of their evaluation of knowledge representation systems. The study was only concerned with the terminological part of knowledge representation systems and used a target representation language of limited expressivity for generating synthetic knowledge bases. The generated knowledge bases, however, are not realistic for Semantic Web-type knowledge bases as we will see from our analysis. Hence, their assumption for class formation [2] is not representative.

1.2 Contribution

In this paper, we provide a systematic approach for the creation of benchmarks for knowledge representation systems. The key characteristic of our approach is that we want to use generating functions to create synthetic ontologies ar, which are derived from structural properties of a given (representative) set of ontologies. If the set of analyzed ontologies is structurally inhomogeneous, clustering techniques are applied to come up with k homogeneous subsets, viz. types of ontologies, for which separate synthetic ontologies can be created. A particular benchmark then consists of several synthetic ontologies representing individual types. Instead of reducing language expressivity to the least common denominator (RDFS in the Semantic Web case), we consider the inability of a particular reasoner to support certain language primitives in our performance evaluation design.

1.3 Limitations

While our approach (with proper adaptation) might be reusable to evaluate other tools relevant to KR-based applications, e.g. editors and visualization tools, our primary focus is set on evaluation of the inference and data processing core of knowledge representation systems. Additionally, we do not consider the identification of a representative list of service requests, which nevertheless are an important aspect in a benchmark. The actual generation of synthetic ontologies is subject of our ongoing research, nevertheless the initial results of our analysis appear to be promising and worth to disseminate.

1.4 Structure of the paper

The paper is structured as follows. Section 2 describes our approach to performance evaluations using benchmarks. Section 3 presents our analysis of the DAML ontology collection, which motivates the necessity for a categorization into several types of ontologies. Section 4 describes the clustering and shows

[2] each class definition is a conjunction containing one or two class symbols (superclasses) zero or one cardinality restriction and zero, one or two value restrictions

how we come up with three categories of ontologies, which are more homogenous. We conclude in Section 5 summarizing our results and giving an outlook to ongoing and future work.

2 Performance Comparisons

We consider benchmarking as the process of performance comparison of two or more reasoners by measurements. A benchmark is the workload used in such measurements. Each performance comparison draws itself on a set of performance criteria or metrics. The choice of the metrics directly depends on the list of services offered by the reasoner.

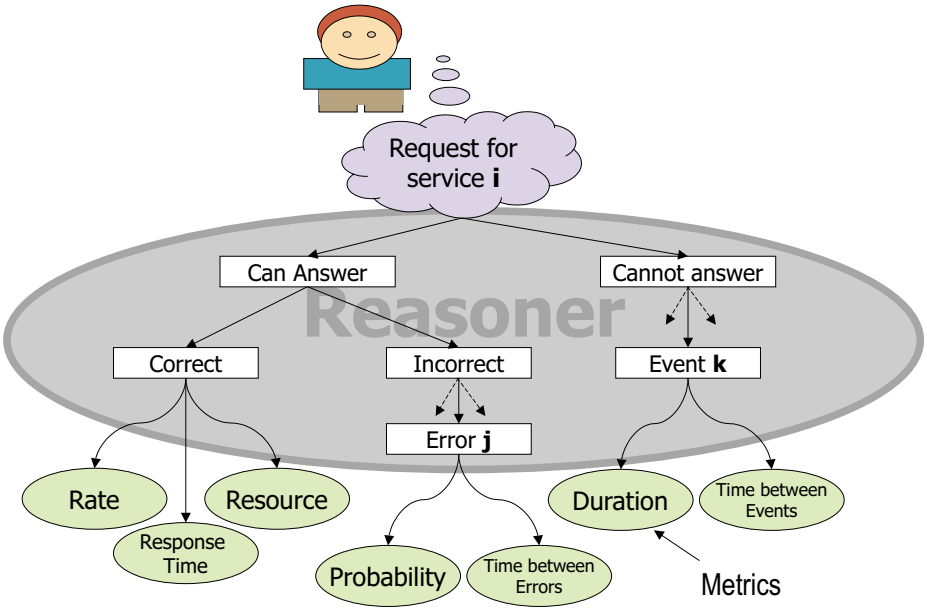

Fig. 1. Services and Metrics in Benchmarks

2.1 Reasoning Services

For each service request several possible outcomes exist (cf. Figure 1). Generally, we can assume that a particular system can either respond correctly, incorrectly or cannot answer the request. A reasoner usually offers query services to interface with the system, several systems also allow update services for manipulation of the knowledge base.

Unlike databases, a reasoner supporting DAML+OIL or OWL, will usually offer several different query services w.r.t. an ontology O. These query services primarily target queries about classes:

1. class-instance membership queries: given a class C,
   (a) ground: determine whether a given individual a is an instance of C;

   (b) open: determine all the individuals in O that are instances of C;
   (c) "all-classes": given an individual a, determine all the (named) classes in O that a is an instance of;
2. class subsumption queries: i.e., given classes C and D, determine if C is a subclass of D w.r.t. O;
3. class hierarchy queries: i.e., given a class C return all/most-specific (named) superclasses of C in the T-Box and/or all/most-general (named) subclasses of C in the T-Box;
4. class satisfiability queries, i.e., given a class C, determine if the definition of C is generally satisfiable (consistent).

There are similar queries about properties, viz. property-instance membership, property subsumption, property hierarchy, and property satisfiability, and also the possibility to check the consistency of the whole ontology / knowledge base.

A single service does not suffice One might want to argue that it is sufficient to measure the performance of the satisfiability, since it is well known, that all queries about classes can be reduced to satisfiability testing. It is important, however, to distinguish different types services, since optimizations can be made for particular services. Naturally the effect of those optimization should be measurable. For example, we might want to measure the performance of a classification service, which can be reduced to several class-subsumption queries (and in turn satisfiability), but reasoners may use different classification algorithms to minimize the number of issued subsumption queries.

2.2 Metrics

For each of the different service requests and their corresponding responses, we can observe a number of metrics. These metrics are later evaluated in the comparison of systems. We may measure successful performance by time-throughput-resource metrics, which measure the responsiveness and productivity and utilization (of system resources) of the reasoner. Notably it is not sufficient to consider response time as the only metric. Some reasoners may be able to respond to requests in parallel, which might lead to a higher throughput. Another reasoner may have a small memory footprint and therefore have a better utilization of system resources. Of course, individual evaluations might consider further metrics.

If the response is incorrect, errors should be returned by the reasoner. Such errors can be classified and it is interesting to determine probabilities for each class of errors and measure the time between such errors. Notably, it is not sufficient to only measure correct performance, since errors are common [6] (even if the reasoning procedures are supposed to be sound and complete).

Several reasons may exist that a reasoner fails to provide an answer at all. Similar to errors, it is sensible to classify failures and determine the probabilities and time between failures for each class. For example, a reasoner may be unavailable due to network errors or software errors or due to lack of support for certain language primitives.

2.3 Workloads

The workload of a reasoner consists of the knowledge base which is loaded by the reasoner and the list of service requests issued by users. We do not consider the identification of a representative list of service requests, which are an important aspect in a benchmark, but concentrate on creating a representative synthetic knowledge base that is subject to user queries.

3 Characterizing Semantic Web ontologies

In order to generate sensible synthetic ontologies, an analysis of available ontologies is necessary, this is the subject of this section.

3.1 Selecting a list of Semantic Web ontologies

For our experiment we chose the DAML.org list of ontologies [4], which contained 247 ontologies at the time of the analysis. Our selection of this set of ontologies is intentional and motivated by the following facts: Firstly, we are not related with the authors of the ontologies in any form. Secondly, most ontologies are created by different people with diverse technical backgrounds (ranging from students to researchers). In this sense, they can be understood as representative for the Semantic Web. Interestingly, many of the ontologies in the library turn out to be just conversions of ontologies, which were initially created in some other representation language. For example, the famous wine ontology is with us since 'Classic' times (for more than 15 years !). This also seems to be a valid assumption for the Semantic Web, which is for sure not created from scratch.

We processed these ontologies using the 1.6.1 version of JENA, we used the Jena DAML API to access the data. The collection contained 189 ontologies in DAML-ONT or DAML+OIL formats, which should be processable by the API in general.

Unfortunately more than 50% of the ontologies contained RDF errors or did not contain valid URIs or did not use RDF(S) namespaces correctly. We did not make any attempt to fix these problems, therefore we could only process 95 ontologies in practise. This results seems shocking, but underlines the need of software that can cope with such errors[3].

The correctness of namespace[4] and RDF usage is, however, not the only relevant property. For example, 21% of the parsable ontologies did not specify the type of properties, viz. whether they are in fact Datatype- or ObjectProperties. In practise, further heuristics need to be applied to make use of these ontologies in reasoners, i.e. deriving the missing type information (e.g. such as done in [2]). Again, we did not make any attempt to fix these problems.

[3] Analogous to HTML browsers, which can cope with all sorts of HTML errors !

[4] A good example for namespace confusion is the NASDAQ ontology (http://www.daml.org/ontologies/342). **Quiz question:** can you spot the inconsistency ?

We based our analysis on the structural properties of asserted information, hence no reasoning was applied. Detailed numbers and sources for the analysis package can be found online[5].

	Average	Std Dev	Median	Min	Max	C.O.V.
Primitive Classes	154,29	1.016,07	5	1	9.795	6,59
Class Expressions	175,20	1.016,39	19	1	9.795	5,80
Restrictions	19,13	44,52	7	-	327	2,33
Enumeration	0,33	1,72	-	-	16	5,26
Set Operation	1,45	8,62	-	-	78,00	5,94
Properties	28,41	43,47	13	-	269	1,53
Object Properties	8,34	31,26	2	-	269	3,75
Datatype Properties	12,57	23,15	4	-	145	1,84
Individuals[6]	29,48	222,32	-	-	2.157	7,54
EquivalentClass	0,73	3,15	-	-	20,00	4,34

Table 1. Average Usage of some language primitives (across all ontologies)

3.2 Average Characterizations

One part of our analysis was concerned with simply counting the usage of certain features. Table 1 summarizes the average usage of language primitives in the ontologies. One important aspect of the summary given in table 1 is that the coefficient of variation (C.O.V.), viz. the ratio of standard deviation and the mean, is high. This shows that the particular ontologies vary tremendously, that is the distribution is highly skewed, hence the median is a more representative characterization of the different numbers than the average.

As we can see primitive classes[7] are the predominant form of class expressions. Different sorts of restrictions are the second most important form of class expressions, interestingly only one ontology actually made use of a single cardinality restriction that would not be expressible in OWL Lite. Seldom enumerations are used, even considering hasValue[8]. Actually, only seven ontologies used set operations to define classes, which are also not available in OWL Lite[9]. Another interesting aspect is that equality is rarely used to define classes (and also rarely used to make properties and individuals synonymous). Also ontologies typically do not contain any individuals. We assume that the pool of individuals will be distributed through the web and is consequently rarely specified together with the ontology.

3.3 Ratios of Primitives

The second part of our analysis concerned the ratio of different primitives in ontologies. The variability of these ratios is smaller than the average counts (cf.

[5] http://kaon.semanticweb.org/owl/evaluation/

[7] By primitive class we denote the atomic named classes that occur on the left-hand sides of subClassOf statements

[8] which can be understood as a syntactically convenient form to express value restrictions with a nominal value

[9] viz. complementOf, disjointUnionOf or unionOf

Ratios	Average	Std Dev	Median	Min	Max	C.O.V.
Primitive/Class Expr.	50%	0,34	39%	6%	100%	0,67
Obj. Prop. / Prop.	24%	0,27	16%	0%	100%	1,12
Dat. Prop. / Prop .	52%	0,38	67%	0%	100%	0,72
Prop. / Prim. Class	3,54	3,89	2,00	-	21,00	1,10
Trans. Prop. / Prop.	0,05	0,10	-	-	0,40	2,30
Obj. Prop./ Prim. Class	0,61	0,83	0,50	-	5,57	1,35
Dat. Prop./ Prim. Class	2,25	3,33	1,00	-	15,75	1,48
Ex. Rest. / Rest.	1%	0,08	0%	0%	65%	5,79
Univ. Rest. / Rest.	48%	0,44	52%	0%	100%	0,91
Card. Rest./ Rest.	34%	0,38	20%	0%	100%	1,11
Rest./Primitive	2,32	2,70	1,50	-	16,00	1,17
Asserted Ind. / Primitive	0,60	4,18	-	-	40,50	6,97

Table 2. Ratio between some language primitives (across all ontologies)

Table 2) since we aggregated relativized numbers. Another effect is of course that numbers do not necessarily add up anymore.

One aspect that can be observed is that the DAML.org library typically contains ontologies and not schemas, since the ratio between Data Properties and Primitive Classes is very low. However, datatype properties are the predominant type of properties. Also some of the ontologies, particularly those with high numbers of classes do not contain any properties, hence the average number of properties per primitive class is very low.

Some 5% of the defined object properties were declared to be transitive. None of the analyzed ontologies contained any functional or inverse functional properties.

Among the restrictions, universal restrictions are predominant, in fact they almost half of all restrictions on average. Typically, a primitive class is further defined by more than two restrictions. Not surprisingly, if we recall our argument for the low number of individuals, at least half of all primitive classes have no direct asserted individuals.

3.4 Distributions of elements in class definitions

The third part of analysis was concerned with determining distributions of the elements contained in class definitions, viz. trying to get answers on questions like: 'How many super-classes does a class typically have ?', 'How many sub-classes ?', and 'How many classes are defined using property restrictions ?'. We did not consider determining distributions for EquivalentClass-statements, as these statements occurred too rarely to come up with a statistically sound, viz. significant, argument.

Figure 2 displays the distribution of sub-classes, super-classes and restrictions per class expression. The y-Axis displays the percentage of class expressions, which have a certain number of subclasses, superclasses or restrictions. The x-Axis represents this number. The last value (15) aggregates all greater numbers, hence the percentage of class expressions is also aggregated. As we can see the distributions are highly inhomogeneous. As our analysis was performed on the syntactic declarations, the semantic properties of description logics, namely

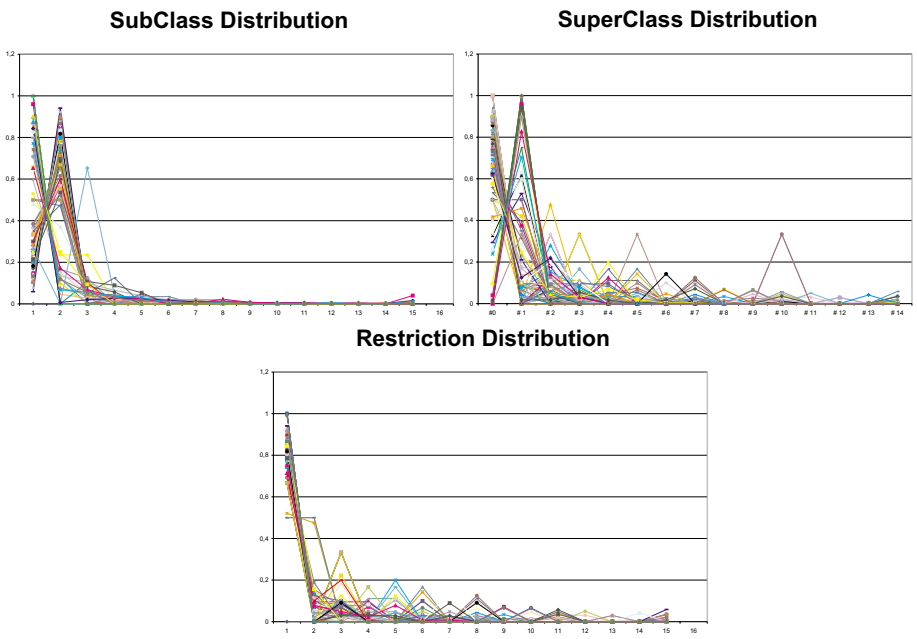

Fig. 2. Distribution of SubClass, SuperClass and Restrictions per Class Expression

that each class is a subclass of daml:Thing are not considered in the distributions, if this were done every class but daml:Thing would have one super-class. Actually, the found ontologies were inconsistent in this respect. Several ontologies redeclared daml:Thing in another namespace (usually the namespace of the ontology). Thing was explicitly assigned as the super-class of a class repeatedly (although this automatically sanctioned by the semantics of the language). Again, these effects were not considered in the analysis.

4 Categories of Ontologies

As discussed before, the distributions of different language primitives is inhomogeneous. However, a quick glimpse on the ontologies suggests that there are different classes of ontologies with a more homogeneous use of those primitives. Thus we applied a clustering algorithm to the data, and indeed found three different clusters.

4.1 Clustering

In order to apply the clustering, a normalization of the data was carried out by using the number of defined classes as denominator. Input values for the clustering algorithm were those language primitives, which were used at least 11 times across all ontologies[10]. The data set consisted of the 95 error-free on-

[10] We chose this number due to the consideration, that a primitive with lower usage, given the small number of ontologies, can only disturb the result.

tologies, which were characterized by 10 attributes, namely Class expressions (95)[11], Primitive Classes(95), Restrictions(69), All Restrictions (48), Cardinality Restrictions (52), Cardinality Restriction covered by OWL Lite (52), Properties (92), Datatype Properties (68), Object Properties (63), and Individuals (19).

We used the WEKA machine learning package to analyze the data, in particular the clustering packages. All attributes have a value range as real value. Hence, we expect unambiguous results from the clustering algorithm, since no transformations need to be applied.

The best results were identified using the k-means[5] clustering algorithm, which initially chooses k random seed points as cluster centroids. It then repeatedly aligns data points to the nearest seed point and calculates the new cluster centers by averaging the assigned data points. This procedure terminates when a certain terminating condition is reached, in our case that no data point is reassigned to another cluster anymore.

A critical decision with k-means is the number of cluster k. We did not evaluate measures like information loss or others in order to define the best number of clusters. We simply evaluated the attributes defined in table 3 for different k and found that $k = 3$ assigns the ontologies in a reasonable way, that is the the coefficient between the improvement of the homogenization (reduction of the COV measure) and the number of clusters k is maximal. More specifically, the clustering allowed us to decrease the c.o.v coefficient to almost half, namely an average of 2,4 in contrast to the 4,5 in 2 using $k = 3$ clusters.

	Average	Std Dev	Median	Min	Max	C.O.V.
Primitive Classes	414	1683	12	1	9795	4,0
Class Expressions	418	1710	15	1	9795	4,0
Restrictions	1,5	4,7	0	0	25	3,1
Properies	39	46	20	0	179	1,2
Object Properties	8	26	0	0	144	3,0
Datatyp Properties	13	25	0	0	108	1,9
Individuals	73	370	0	0	2157	5,1

Table 3. Average Usage of some language primitives (across clustered ontologies(C1))

4.2 Cluster contents

A closer examination of the ontologies assigned to the different clusters reveals that the clusters correspond more or less to three types of ontologies. The largest cluster of ontologies seems to contain ontologies of taxonomic or terminological nature. The ontologies are characterized by few properties and a large number of classes, cf. Table 3.

The second largest cluster contains description logic-style ontologies. This cluster is characterized by a high number of axioms per class and a low number of primitive classes. These ontologies also contain a very high number of restrictions and properties (especially datatype properties), however almost no individuals.

The third cluster contains database schema-like ontologies. The ontologies are medium size containing on average 65 class expressions and 25 properties.

[11] The value in brackets specifies the number of ontologies, where values occured

This cluster is more inhomogeneous as indicated by high standard deviations per primitive.

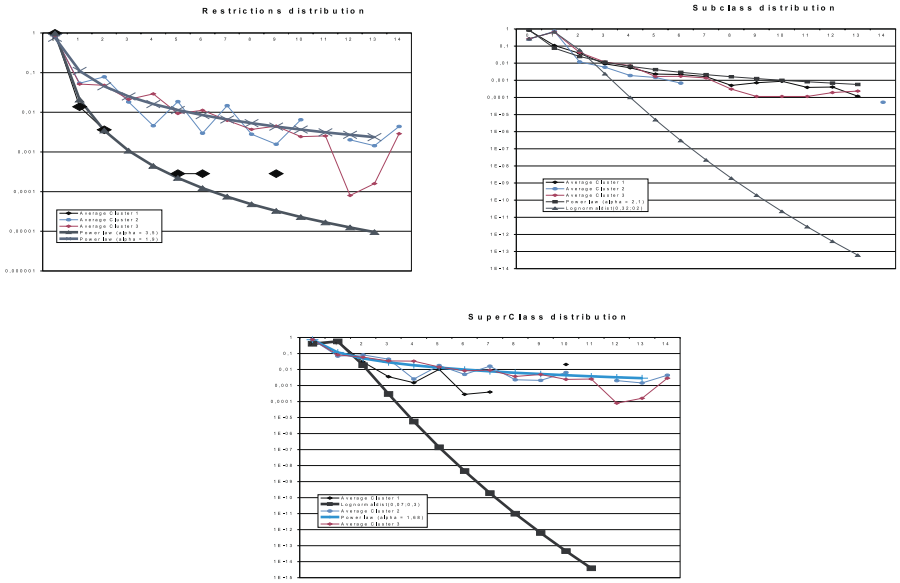

Fig. 3. Average Distribution of SubClass, SuperClass and Restrictions for the Clusters with the estimated distributions (y-axes is log scale)

4.3 Feature Distributions

Having clustered the ontologies into more consistent classes, we now have a look at the distributions of certain features in the taxonomic cluster and the database-like cluster. Again, due to lack of space, we will not look at all feature combinations but rather examine two representative features, namely restrictions per primitive class (database-like)[12] and the distribution of subclasses per class expression (cf. Figure 3 (taxonomic)).

Restrictions In particular we had a look at the distribution of restrictions across classes in the different clusters. In average 1,5[13] (C1), 26[14] (C2) and 30[15] (C3), 17 restrictions are defined in each ontology. Hence, 0.004 (C1), 0.6 (C2) and 0.63 (C3) per class. We compared the observed distributions with the expected values of parameterized distribution functions, in particular the exponential distribution and the power law distribution [9]. Intriguingly the distributions of restrictions closely corresponds to power law distributions with $\alpha = 3,5$ (C1),

[12] The absolute number of restrictions in the taxonomic case is to small to analyze the data expecting significant results.
[13] standard deviation of 4,8
[14] standard deviation of 59
[15] standard deviation of 48

$\alpha = 1,9$ (C2) and $\alpha = 1,8$ (C3). This argument is supported with a confidence value of 99,9 % (using the χ^2-Test). α was estimated to fit the average of the observed distribution. In this case the estimated standard deviation differs at most 16% (C1) from the actual standard deviation. In case of cluster C3 with the most restrictions the difference is just 2% which underlines the argument for a power law distribution.

Sub Classes per Class Considering the distribution of sub classes per class, we found that in the taxonomic-like cluster (C1) each class had 0.30 subclasses with a standard deviation of 0.86. At this point we want to recall, that we did not apply any reasoning to the data set. This would probably alter the figures a bit. We found that the distribution of sub classes per classes also follows a power law distribution with $\alpha = 2,2$. As in the case of restrictions, the argument is supported with a confidence value of 99%. However, the estimated standard deviation differs 40% from the actual observation. The other two clusters (C2, C3) seem to follow a lognormal distribution for the occurrence of sub classes for the first two classes, but than the distribution seems more like a power law distribution. However, a look at the distributions for Super Classes per Class shows an inverted picture, with clusters C2, C3 following a power law distribution and cluster C1 a lognormal one.

5 Conclusion

We provided a systematic approach for the creation of benchmarks for knowledge representation systems and presented the results of the first step in benchmark creation - the analysis of available data. Using our analysis of the DAML.org library, we can use generating functions, e.g. an exponential distribution with the calculated mean for the distribution of restrictions, to generate ontologies for benchmarking, that correspond structurally to real-life ontologies.

Our analysis shows, that benchmarks have to consist of several types of ontologies, since the set of analyzed ontologies would otherwise be too inhomogeneous to derive parameters. As our analysis showed, 3 types of ontologies can generally be identified. For each type of ontologies, high confidence values for the generator functions could be shown.

Our future work is concerned with implementation, viz. an online web service to generate synthetic ontologies, and with deriving realistic workloads for modelling user requests. To this extend we plan to monitor existing ontology-based applications, e.g. the OntoWeb portal and the portal of our institute.

References

1. P. Balsinger and A. Heuerding. Comparison of theorem provers for modal logics - introduction and summary. In *Proc. of Tableaux'98*, pages 25–26, 1998.
2. Sean Bechhofer, Raphael Volz, and Philip Lord. Cooking the Semantic Web with the OWL API. In *ISWC 2003*, Sanibel Island, Florida, USA, October 2003.
3. Jeremy Caroll and Jos De Roo. OWL Web Ontology Language Test Cases. Internet: http://www.w3.org/TR/owl-test/, May 2003.

4. DAML.org Ontology Library. Internet: www.daml.org/ontologies, As of July, 25th 2003.
5. J. A. Hartigan and M. A. Wong. Algorithm AS136. A K-means clustering algorithm. *Applied Statistics*, 28:100–108, 1979.
6. J. Heinsohn, D. Kudenko, B. Nebel, and H.-J. Profitlich. An Empirical Analysis of Terminological Representation Systems. *Artificial Intelligence*, 68(2):367–397, August 1994. 1994.
7. A. Heuerding and S. Schwendimann. A benchmark method for the propositional modal logics k, kt, s4. Technical Report IAM-96-015, University of Bern, Switzerland, October 1996.
8. I. Horrocks and P. F. Patel-Schneider. DL systems comparison. In E. Franconi, G. De Giacomo, R. M. MacGregor, W. Nutt, C. A. Welty, and F. Sebastiani, editors, *Collected Papers from the International Description Logics Workshop (DL'98)*, pages 55–57. CEUR, May 1998.
9. G. K. Zipf. *Selective Studies and the Principle of Relative Frequency in Language*. Harvard University Press, 1932.