# Empirical Investigation of Subsumption Test Hardness in Description Logic Classification

Nicolas Matentzoglu, Bijan Parsia, and Uli Sattler

The University of Manchester
Oxford Road, Manchester, M13 9PL, UK
{matentzn,bparsia,sattler}@cs.manchester.ac.uk

**Abstract.** Recently, modular techniques have been employed for optimising Description Logic reasoning, specifically to enable incremental reasoning and improve overall classification time. Properties of locality-based modules strongly suggest that classifying a module of an ontology should be significantly easier than reasoning in the whole ontology, either due to subsumption test avoidance (traversal) or reduction of subsumption test time. However, we observed in previous work that neither it is generally true that modular reasoning techniques have a reliable positive effect, nor even that the classification time of a module is less than or equal to the classification time of the whole ontology. One possible explanation for the latter could be that counter-productive optimisations are triggered within the reasoner when dealing with the sub-module, and thus individual subsumption tests get harder when parts of the ontology are missing. The goal of this paper is to understanding the contribution of subsumption tests to the hardness of classification. The contribution is twofold: (1) We analyse the impact of subsumption test hardness on DL classification by analysing a well known corpus of ontologies, and (2) we present a novel approach based on modularity to robustly detecting subsumption tests that are *too hard*.

**Keywords:** classification,ontologies,benchmarking,modular reasoning

## 1 Introduction

Reasoning in popular, very expressive Description Logics (DL) is very difficult (e.g., $\mathcal{SROIQ}$ is N2ExpTime-complete) [12]. Perhaps surprisingly, modern reasoning systems suitable for the entirety of OWL 2 DL (essentially a notational variant of $\mathcal{SROIQ}$) such as FaCT++ [20], Pellet [18], HermiT [5] and recently Konclude [19] generally perform well against real ontologies. However, due to the poor performance in some (often important) cases, the quest for optimisations is ongoing. The need to empirically validate such optimisations stems from the sheer complexity of reasoner architectures. Worst case complexity analysis and its variants do not account for the high variability of classification times of real ontologies. Modern reasoning systems have to accommodate multiple reasoning services and also tend to implement a wide range of optimisation techniques that

might affect each other. Various sources of non-determinism, mainly traversal (subsumption test order) and or-branch exploration of a tableau further complicate the situation. Statistical methods such as linear regression [16] tend to be only precise for trivial cases, and are limited in their explanatory richness. Currently, principled benchmarking provides the only way to creating detailed characterisations of DL reasoning performance.

Using locality-based modules to optimise Description Logic classification experienced a resurgence in recent years [15, 21]. Intuitively, breaking the input problem into smaller pieces, reasoning over those pieces separately, then recombining the results is appealing. From a naive perspective, it seems obvious that a straightforward way of dealing with high worst case complexity is to keep the inputs very small. Furthermore, if there are especially difficult parts of the ontology, perhaps they can be isolated to reduce their impact. In practice however, modular reasoning techniques do not always improve the performance of classification [6]. In fact, they can drastically impair performance, making it a hit and miss game to chose between a modular reasoner (e.g. MORe-HermiT, Chainsaw-JFact) and its monolithic counterpart (e.g. HermiT, JFact). These cases can often be due to various kinds of overhead induced by modular reasoners (e. g., extracting the modules might take longer than classifying the whole) or redundancy introduced by the mostly unavoidable and often significant overlap between the various modules extracted. In a preliminary set of experiments [13] we observed a curious effect: Not only are there cases where there are individual subsumption tests that can be, often significantly, harder in a module extracted by a modular reasoner than in the whole ontology, but we could observe that there are occasionally modules whose classification time exceeds that of the entire ontology $\mathcal{O}$ it was extracted from.

The **goal** of this paper is to understanding the contribution of subsumption tests to the hardness of classification. The **contribution** is twofold: (1) We analyse the impact of subsumption test hardness on DL classification by characterising a well known corpus of ontologies, and (2) we present a novel approach based on modularity to robustly detecting subsumption tests that are potentially *too hard*. As a result, we re-confirm the almost 20 years old results by Horrocks [11] that subsumption tests are generally rather easy. We also isolate counter-intuitive instances that, however, are often likely to be the consequence of the surprising degree of observed stochasticity in the classification process.

## 2 Background

Understanding the experimental design and methodology presented here does not require more than a cursory understanding of the syntax, semantics, and proof theories implemented. A **reasoner** is a program that offers key logical services usually involving deriving entailments in an ontology. The most prominent families of reasoning algorithms for description logics are tableau (incl. hyper-tableau) and consequence-based. In our work, we are mainly concerned with tableau algorithms. **Logical services** are, among others, classification,

consistency checking, simple entailment checking, explanations and instance realisation. The reasoners under investigations in this paper are designed to implement key reasoning services for the Web Ontology Language (OWL), most importantly classification and consistency checking. Given an ontology (a set of axioms) $\mathcal{O}$, the signature of an ontology $\widetilde{O}$ is the set of non-predefined entities (classes, individuals, object properties, data properties) appearing in the axioms in $\mathcal{O}$. We use $CT(\mathcal{O})$ (classification time) and $CT(\mathcal{M})$ respectively to denote the time of computing the set of atomic subsumptions (i.e., statements of the form $A \sqsubseteq B$ where $A$ and $B$ are predicates in the signature) or *classification* of $\mathcal{O}$. For brevity, we refer to overall classification time as OCT and subsumption test time STT.

While subsumption testing, and therefore classification, is in theory intractable, highly optimised reasoners do fairly well in practice. The observed efficiency despite the worst case complexity is in principle down to four factors. (1) Real ontologies are bounded in size and expressivity. That means that the theoretically hard cases might never or rarely occur. (2) Many ontologies fall into tractable fragments of OWL, and can be classified using efficient polynomial algorithms such as the ones from the family of consequence-based algorithms. (3) The last 20 years brought a plethora of different optimisations to make *satisfiability checks* (and therefore subsumption tests) *easier* [11, 3]. (4) Very efficient algorithms were developed to avoid the vast majority of subsumption tests altogether [1, 4, 17].

The particular flavour of "logically respectable" **modules** we use are based on *syntactic locality* [10]. Current modular classification approaches use so-called $\bot$-modules (bottom-modules) which have a number of desirable properties: (1) Being based on syntactic locality, they are relatively cheap to extract and are reasonably compact and exact. (2) If $\mathcal{O} \models A \sqsubseteq C$ then for any given $\bot$-module $\mathcal{M}_\bot$, of $\mathcal{O}$ where $A \in \widetilde{\mathcal{M}_\bot}$, $\mathcal{M}_\bot \models A \sqsubset C$ (were $C$ is an arbitrary expression over the signature of $\mathcal{O}$). Thus, $\bot$-modules are classification complete for their signature with respect to their parent ontology. Hereafter, we will use $\mathcal{M}$ to refer to a syntactic locality based $\bot$-module.

Very recently, reasoner developers have started to **utilise modularity for classification**. They either are (1) using modules for incremental reasoning [9] or (2) using modules to improve classification time [15, 21].

Most OWL Reasoner benchmarks, especially those focused on classification, determine how long it takes the reasoner to execute the service for a given input. Few benchmarks distinguish between different stages outside the actual reasoning. We propose a model of monolithic reasoning that distinguishes the following five stages for the process of classification: (1) Preprocessing, (2) initial consistency check, (3) pre-traversal optimisations, (4) traversal, (5) postprocessing. The reasoning systems we analysed in our framework all follow that model, and we believe that most classifiers do. The second core aspect of the framework is the *recording of subsumption tests*. We restrict ourselves to calls to the Tableau implementation of the reasoner, and ignore subsumptions determined by other means (e.g. nested consequence-based procedures). From an implementation per-

spective, the framework currently has to be hard wired into the code (a single static Java class). While we did this ourselves for the Java-based systems in our study, we collaborated with the developer of FaCT++ in order to extend the interface to merely flushing out textual information that we then later parsed back into our analysis framework. All measurements reflect wall clock time, to avoid the confusion provided by multi-threaded implementations. We also believe that wall clock time more realistically reflects what end-users are interested in, despite being a considerable source of experimental error.

## 3 Related Work

Attempts to understand DL reasoning performance are, up until today, rarely systematic or comprehensive. Recently, the ORE reasoner competition tries to establish the methodological foundations for more reliable comparisons [6] between different reasoners and across a range of different reasoning services. OWL Reasoner benchmarks have been conducted for varying purposes, for example (and most prominently) guiding end-users for selecting appropriate reasoners for their problem [2, 6] or understanding reasoning or the state of reasoning in general [7]. Dentler et al. [2] conduct a principled investigation to identify suitable criteria for choosing an appropriate reasoner for EL ontologies. In our work, we are interested in mapping out subsumption test hardness during full classification across reasoner-ontology pairs (phenomenological characterisation) and the potential of modularity to pinpoint counter-intuitive cases (i.e. harder tests in a sub-module). Most benchmarks conduct an only semi-principled dataset selection: Even carefully executed benchmarks such as Dentler et al. [2] usually cherry pick a set of *somehow* relevant ontologies. Few works sample from existing corpora or the web, and only Gonçalves et al. [7], to the best of our knowledge, deal with corpora larger than 500 ontologies. In practice, the current de facto gold-standard corpus for ontology experimentation is BioPortal [14], which also provides a well designed infrastructure to obtain an interesting range of biomedical ontologies programatically. We are using a snapshot of BioPortal in this work. As far as we know, no benchmark to date has investigated *subsumption testing during classification* across reasoners in a principled manner. However, various benchmarks have investigated the effect of certain optimisations on subsumption test avoidance [4]. While the literature on classification optimisation and reasoning is vast, little progress has been made in understanding classification hardness of real ontologies, both empirically and formally.

## 4 Subsumption Test Hardness

Before we explain our model of subsumption test hardness, we first provide some operational definitions. The phenomenon under investigation is **subsumption test hardness in the context of classification**. A **subsumption test** is a question asked by the reasoner to determine whether $A \sqsubseteq B$. The **subsumption test hardness** is the time it takes to compute the answer, operationalised as

wall-clock time. In this work the answer to a test is either yes or no. Note however, that for any implementation (1) more than just a binary answer will be provided (i.e., cached models, derived subsumptions) and (2) no guarantee is given that the answer is correct (accidental unsoundness). "In the context of classification" means that we are not exploring individual "cold" tests, i.e. letting the reasoner compute whether $A \sqsubseteq B$ for any $A, B$ from outside the classification process, because we want to understand the contribution of subsumption testing to classification as a whole, with all the optimisations involved.

Our **model of subsumption test hardness** with respect to sub-modules is based on the following intuition: Given a *positive* subsumption test $\mathcal{ST}$, it should be the case that for every two modules $\mathcal{M}_1$,$\mathcal{M}_2$ with $\mathcal{M}_1 \subset \mathcal{M}_2 \subset \mathcal{O}$ in which the subsumption test is triggered, the hardness of $\mathcal{ST}$ always stays the same. The reason for that are module properties: every justification for an entailment is part of every module that entails it. Thus, every way that the entailment holds is contained in the module, no "new" information about the entailment exists in the rest of the ontology. Intuitively, additional "stuff" can make it *harder* to figure out the entailment, but not make it easier. This makes this metric a possible indicator of counter-productive optimisations: If we find that $\mathcal{ST}_{\mathcal{M}_2}$ is harder than $\mathcal{ST}_{\mathcal{M}_1}$, we might conclude that the reasoner is doing some unnecessary extra work in $\mathcal{M}_2$; if $\mathcal{ST}_{\mathcal{M}_1}$ is harder than $\mathcal{ST}_{\mathcal{M}_2}$, there is a possibility that a counter-productive optimisation may have been triggered. Only the second case is truly pathological: A test should never get harder when *irrelevant* axioms are removed from the ontology. The first case might simply occur because if $\mathcal{M}$ grows, it gets harder to identify the irrelevant axioms. One possibility to explain both cases may be the inherent stochasticity of classification as implemented by current OWL Reasoners. For example, a random factor might (for example by changing the test order) simply shift the load of $\mathcal{ST}$ in $\mathcal{M}_1$ to another subsumption test $\mathcal{ST}_2$ that consecutively makes $\mathcal{ST}$ easier. Another reason for a test becoming easier in a sub-module might be the exploitation of partial results from negative tests (e.g. caching).

Our empirical investigation of subsumption tests has two parts: (A) a broad characterisation of the landscape of subsumption testing and (B) an in-depth characterisation of non-trivial subsumption tests. We treat a test as non-trivial if it takes longer than 100 ms . The first **part A** will attempt to answer the following questions: What is the impact of subsumption testing on reasoning performance in general (RQ1)? How many tests are positive or negative and how do they differ in hardness (RQ2)? How hard are real tests actually (RQ3)? We address RQ1 by breaking down how many ontologies require the reasoner to trigger tests, and provide an analysis of the impact of these tests on the overall classification time. The impact is quantified as the contribution to the overall classification time in percent. RQ2 will be answered statistically by observing the distribution of positive and negative tests across the hardness bins. RQ3 will be addressed by binning subsumption tests and ontologies according to their hardness as described in the end of this chapter, broken down by reasoner. The motivation for this first part is twofold: (1) we want to get a feeling for the

hardness of real subsumption tests during classification in the wild, (2) we want to identify relevant cases for the in-depth characterisation of non-trivial cases (B).

**Part B** serves as an in-depth characterisation that attempts to address questions related to the general stability of the measurements (**intra-module**) and the effect of modularity (inter-module). **The following questions will be answered from the intra-module perspective.** Is subsumption test hardness a stable phenomenon (RQ4)? This is important in order to judge how reliably we can trace a single subsumption test through different sub-modules of an ontology, and may also give a warning sign for triggered non-determinism, for example in the case that a test appears or disappears given a particular ontology-reasoner pair across runs. We will address this problem mainly by looking at the *coefficient of variation* (COV) of subsumption test hardness, a standardized measure of dispersion of a distribution defined as the ratio of the standard deviation to the mean, across different runs. What are the reasons for instability (RQ5)? We will not conclusively try to answer this problem, but we will collect some evidence for stochasicity by looking at intra-module variation of test counts, a strong indicator of non-determinism. Evidence against measurement error will mainly come from the mean coefficient of variation of test hardness across all the reasoner - module pairs.

The next questions will be asked from the **inter-module perspective.** Does modularity change the hardness of tests (RQ6)? In order to answer this question, we will classify tests by analysing how modularity effects their hardness. This happens as follows: We identify all super and submodule combinations $\mathcal{M}_1, \mathcal{M}_2$ as described earlier. For each test triggered in both $\mathcal{M}_1$ and $\mathcal{M}_2$, we determine: (1) was the effect positive on average (across runs), (2) what was the magnitude of the effect and (3) was the effect stable? We define **stability of an effect** as follows: given a subsumption test $\mathcal{ST}$ that occurs in two modules $\mathcal{M}_1, \mathcal{M}_2$ with $\mathcal{M}_1 \subset \mathcal{M}_2$, and two sets of measurements $MEAS(\mathcal{ST}_{\mathcal{M}_1})$ and $MEAS(\mathcal{ST}_{\mathcal{M}_2})$ (a) measurements $ME \in MEAS(\mathcal{ST}_{\mathcal{M}_1})$ are either all harder or all easier than measurements $ME \in MEAS(\mathcal{ST}_{\mathcal{M}_2})$ (strong stability) or (b) the overlap of the ranges of $MEAS(\mathcal{ST}_{\mathcal{M}_1})$ and $MEAS(\mathcal{ST}_{\mathcal{M}_2})$ is less than 10% of the range of $MEAS(\mathcal{ST}_{\mathcal{M}_1}) \sqcap MEAS(\mathcal{ST}_{\mathcal{M}_1})$.

For improved readability of results, we group subsumption test hardness into the following bins: Very Hard (more than 100 seconds), Hard (>10 sec), Medium Hard (>1 sec), Medium (>100 ms), Medium Easy (>10 ms), Easy (>1 ms), Very Easy (>100 $\mu$s.), Trivial (<100 $\mu$s).

## 5 Experimental Design

We have conducted three separate experiments, each addressing a distinct part of our overall problem: (1) The characterisation of subsumption test hardness in the context of classification across a well known corpus, (2) the in-depth analysis of a subset of cases for variability and counter-intuitive cases and (3) the causal investigation into pathological cases.

### 5.1 Corpus, Reasoners, and Machines

We conducted our study on a corpus of 339 OWL API (3.5.0)-parsable BioPortal ontologies, obtained through the BioPortal REST Services[1] (January 2015 snaphshot). All ontologies were serialised into OWL/XML, with merged imports closure. A minimum amount of repair (injecting missing declarations, dropping empty n-ary axioms, etc.) was applied to ensure that trivial violations do not impair DLness. An overview of the corpus features can be seen in chart 1.

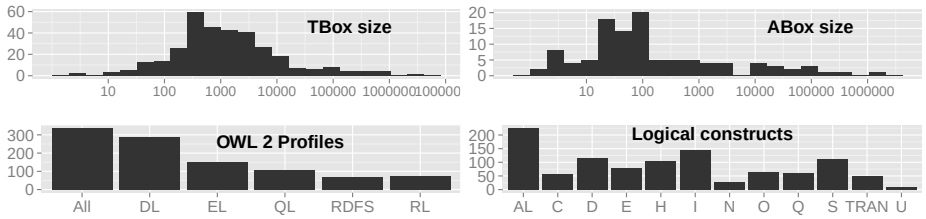


**Fig. 1.** Corpus Metadata. Top: histogram of axiom counts (y-axis: ontology count, x-axis: axiom count. Bottom: y-axis: number of ontologies exhibiting respective feature.)

For all our experiments, we use four state-of-art OWL reasoners that implement the OWL API OWLReasoner interface: HermiT 1.3.8, Pellet 2.3.1, JFact 1.2.3 and FaCT++ 1.6.3. All four are among the most heavily used reasoners for OWL 2 DL. The reasoners have been modified for the benchmark in the following way: (1) A timestamp is recorded when a stage according to our reasoner stage model is entered; (2) When a subsumption test is conducted, the start and end timestamps, the sub and super class under consideration and the result of the test are recorded. The recorder is a lightweight static Java class. In order to mitigate performance impairments through memory overhead, we use a buffered file writer for the subsumption test data and record the overhead writing to it. While we can use this approach to compare results for each reasoner, interpretation of comparisons between reasoners might be misleading due to implementational details. For example, methods that test for subsumption and ultimately satisfiability may be heavily nested. In order to chose where exactly to measure, we either asked the developers directly (JFact, FaCT++) or were guided by benchmarking code already present (progress monitors for debugging, HermiT and Pellet). Because we are interested in real life behaviour, we allowed the reasoner to fall into states like the deterministic part of HermiT for Horn-SHIQ or Pellets internal EL-Reasoner. That said, we cannot claim to measure `all` subsumption tests a reasoner does, because subsumptions are determined during consequence-based approaches or HermiT's deterministic reasoning as well. We can, however, establish a lower bound and are confident that we capture the vast majority of

[1] `http://data.bioontology.org/documentation`

the hard tests, because the sum of test times occasionally account for almost 100% of the OCT for all reasoners.

A set of four equal-spec Mac Minis with Mac OS X Lion 10 (64 bit), 16 GB RAM and 2.7 GHz Intel Core i764-bit was used for the benchmarking. Every single classification was done in a separate isolated virtual machine (Java 7, -Xms2G, -Xmx12G). In order to reduce potential bias induced by run order (unaccounted for background processes kicking in, runtime optimisations), we fully randomise the run order and evenly distribute the experiment run jobs across the four machines.

### 5.2 Experimental Pipeline

For the first experiment we execute a single run of all reasoners across the entire corpus (4 reasoners on 339 ontologies), with a timeout of 60 minutes per run. Due to technical details, the timeout is a lower bound and might not be triggered until some minutes later. Note that we include every ontology in the corpus, including the ones not strictly in OWL DL (53). The reason for that is that these ontologies do, if we like it or not, form part of the landscape, and reasoners are used on them. The main sources of violations are uses of reserved vocabulary (37% of all violations across the corpus), illegal punning (32%) and uses of datatypes not on the OWL 2 datatype map (11%).

For the second experiment, we select a set of reasoner-ontology pairs for which, according to the results of experiment 1, at least one test was measured that was harder than 100 milliseconds. Because of the various claims we have with respect to modules, we also excluded ontologies that do not fall under OWL 2 DL. *Runtime limitations* (both for the data gathering and the analysis) forced us to focused on those ontologies with *less than 5200 tests in total* (5000 would have left the sample without a case for FaCT++). For this experiment, we first obtain random cumulative subsets from the ontologies in our narrowed down sample, similar to Gonçalvez et al. [8], with 16 slices. In a nutshell, given the set of logical axioms the ontologies is comprised of, we obtain a random 1/16th of the axioms, serialise this subset, add another randomly drawn 1/16th from the remaining axioms to the first, serialise them together, and then iteratively grow each consecutive subset until the final set is the whole ontology. From the signature of the each subset sampled, we obtain the $\bot$-locality module using the OWL API module extractor. Module properties ensure that, given subset $S1 \subset S2$, $\mathcal{M}_{S1} \subset \mathcal{M}_{S2}$. The module of 16/16th, $\mathcal{M}_{\widetilde{\mathcal{O}}}$, corresponds to the whole ontology. We call this nested set of modules a **path**. Note that the modules are usually considerably larger than their respective subsets, which will give us a good sample of relatively large modules with hopefully hard subsumption tests. Each of the modules obtained is classified three times (i.e., three independent runs) by each reasoner. Given a path $\mathcal{M}_1 \subset \mathcal{M}_2 \subset \mathcal{M}_n$, we call $\mathfrak{P}$ the set of all pairs $\mathcal{M}_i, \mathcal{M}_j$ with $i<j$.

## 6 Results

Supporting materials, data sets and scripts can be found online[2]. Percentages in this section are subject to appropriate rounding.

### 6.1 Subsumption Test Landscape

Out of the 1356 attempted classification runs (4 reasoners and 339 ontologies), 1136 (85%) completed successfully. 322 ontologies were dealt with by at least one reasoner (95%) within the 60 minute timeout. Reasons for failure include hitting the timeout, unsupported datatypes (FaCT++), and lack of DLness (mainly HermiT). From the 322 ontologies successfully processed, 186 did not have any subsumption tests measured by any of the three reasoners. By reasoner, FaCT++ did not test in 177 cases, HermiT in 189, JFact in 191 and Pellet did not fire a subsumption test during 218 successful classifications. For the remaining 136, at least one reasoner conducted a subsumption test as described in Section 5.1. In table 6.1 we can see that ontologies with tests are generally considerably harder (consider mean and third quartile).

| | Count | Min. | 1st Qu. | Median | Mean | 3rd Qu. | Max. |
|---|---|---|---|---|---|---|---|
| O with tests | 136 | 0.051 | 0.548 | 1.503 | 97.490 | 4.329 | 7,575.000 |
| O without tests | 186 | 0.046 | 0.211 | 0.418 | 40.140 | 1.069 | 8,014.000 |

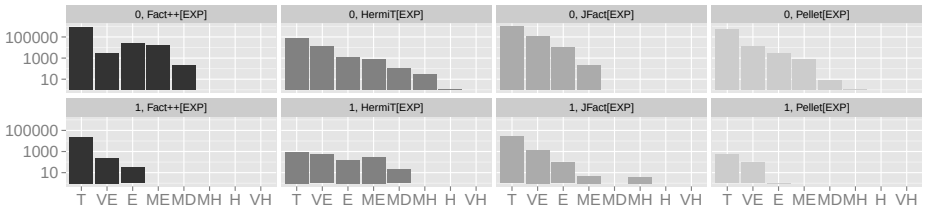


**Fig. 2.** Counts of subsumption tests for each hardness bin by reasoner (log scale).

In Figure 4 we can observe the impact of subsumption tests on the overall classification time (OCT), broken down by reasoner. One interesting observation is that most positive tests are of only trivial hardness, while negative tests are generally harder. The OCT includes all stages described in section 2. While subsumption testing dominates the OCT only in a few cases, it occasionally accounts for more than 80%. Very rarely we can observe a single test accounting for more then 10% of the OCT. The maximum impact for a single test by Pellet is 11.3%, HermiT 23.1%, JFact 24.8% and FaCT++ 9.2%. The distribution of subsumption test hardness across all runs according to our hardness scale (Sec. 4) is shown in Figure 2 (left). Figure 3 shows how many ontologies contain tests

[2] `http://owl.cs.manchester.ac.uk/publications/supporting-material/subtest-hardness-in-classification/`

of a particular hardness. The most important observation to make here is the rarity of ontologies with tests that take longer than a second (medium hard bin and above).

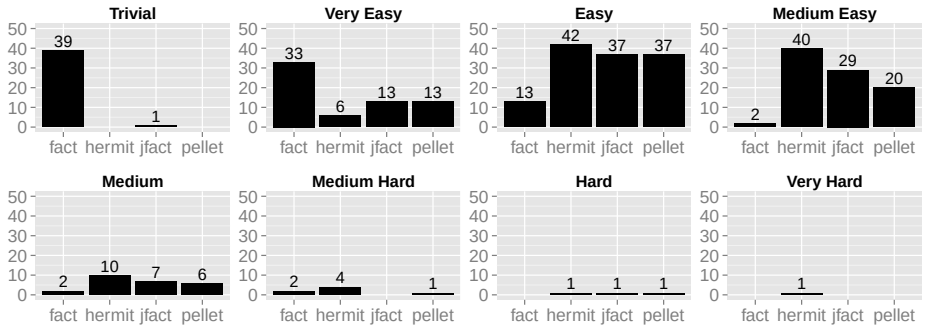


**Fig. 3.** Ontologies in each hardness category. The x-axis represents the number of ontology in each bin. Bin classification according to hardest subsumption test.

| | Min. | 1st Qu. | Median | Mean | 3rd Qu. | Max. | Pos | Neg |
|---|---|---|---|---|---|---|---|---|
| FaCT++ | 2 | 46 | 71 | 7,519 | 111 | 2,352,000 | 24,286 | 905,011 |
| HermiT | 48 | 418 | 481 | 17,390 | 570 | 198,900,000 | 1911 | 88387 |
| JFact | 1 | 48 | 88 | 1127 | 169 | 45,920,000 | 28103 | 1100972 |
| Pellet | 23 | 175 | 246 | 825 | 365 | 35,060,000 | 634 | 522592 |

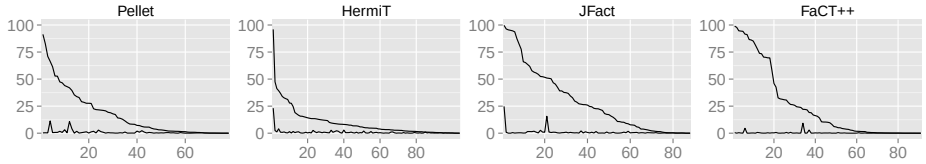


**Fig. 4.** Impact of SST on classification time by reasoner in %. Low line: hardest individual test; high line: sum of all tests; x-axis: ontologies (by index).

### 6.2 In-depth Characterisation

From the previous experiment, according to the process detailed in Section 5.2, 1 ontology was selected for FaCT++, 10 for HermiT, 1 for JFact and 4 for

Pellet. The full ontologies have OCT's ranging from 7.31 seconds to 1211.00 seconds (median: 95.87, mean: 179.50). Out of the 768 (16 modules per ontology, 3 runs per module) attempted classifications (timeout 60 minutes), 741 (97%) successfully terminated. Out of the possible 256 modules (16 modules, 16 ontology-reasoner pairs) across the entire set, we obtain 251 records from the **intra-module analysis**, 242 out of which were obtained from three distinct measurements, 6 are comprised of two distinct measurements and 3 by only one. Since we are interested in observing variability, we discard the latter 3 and stick with 248 partially or fully complete records. Variability is determined using the coefficient of variation (COV). From the module perspective, we look at three distinct sources of variation: overall classification time OCT, sum of all subsumption test times (SUMST) and the total number of tests conducted (CTT). Across modules, only 1 module OCT varies by more than 30%, 10 by more than 20% and 17 by more than 10%. The module with the worst variation corresponds to a module taken from a 3/16th of the CAO ontology, classified by Pellet (min=104.25 sec, max=215.08 sec). A more detailed picture of the overall variation can be taken from Figure 5. In terms of test count, the variation is surprisingly large. 184 out of 248 cases (75%) show differences in the number of test measured across runs. Only 63 (25%) do not vary at all. 16 modules vary by as much as 20% in the number of subsumption tests.

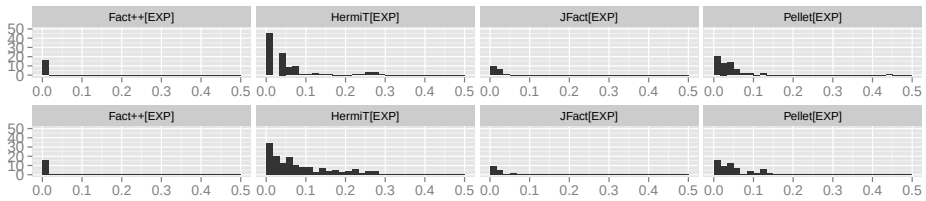


**Fig. 5.** Histogram of variation (COV) by reasoner. Top: OCT, bottom: SUMST

Across all 251 modules, we measured the hardness of 204,286 distinct subsumption tests. Only 92% of the tests are measured more than once and we discard the rest. As can be seen in Figure 6, the coefficient of variation is generally log10-normally distributed (here reported in percent rather than in proportions of 1), but varies considerably across reasoners. On average measurements deviate as much as 13.22% for Pellet, while measurements for HermiT and JFact deviate 8.90% and 8.00% respectively, and FaCT++ only 2.33%. The maximum variation for any test measurement for Pellet is 172.65%, for HermiT 170%, for FaCT++ 154% and for JFact 32%.

For the **inter-module analysis**, we sampled 10 sub-module super-module pairs from $\mathfrak{P}$ from the 120 possible combinations as described in Section 5.2. For result stability, we excluded 2 from the resulting 160 pairs that had only a single measurement for either the sub or the super-module, and continued with 158. Figure 7 shows the overall changes in measurement times across pairs by rea-

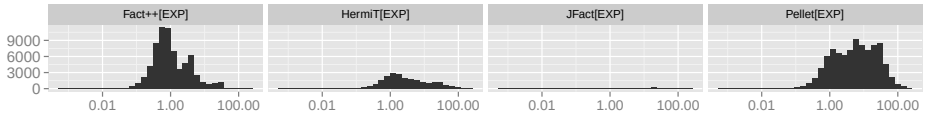


**Fig. 6.** Histogram of variation (COV) of subsumption test measurements by reasoner. Mind the log scale.

soner. Bin membership is determined as follows. Given a pair $<\mathcal{M}_1, \mathcal{M}_2> \in \mathfrak{P}$ we look at the change from either the $CT(\mathcal{M}_1)$ to $CT(\mathcal{M}_2)$ (module perspective) or the change from a subsumption test $\mathcal{ST}_{\mathcal{M}_1}$ to $\mathcal{ST}_{\mathcal{M}_2}$ (subsumption test perspective). Every pair of measurements has a tendency, a magnitude and a degree of stability. The tendency *easier* (mean hardness change less than 5%) denotes that a test is easier in the super-module (potentially pathological), *harder* (mean hardness change more than 5%) the reverse, and neutral means the mean measurement difference does not change by more than 5%. High magnitudes are changes above 50%, medium above 5% and low below. A stable effect can be clear cut, high or low 4. Neutral cases have high stability if both sets of measurements have a variation coefficient less than 5%. From the module perspective, the main observation to be made here is that there are 8 cases in the set where the sub-module is harder than the super-module and 38 where there is no significant change in hardness (less than 5% change). Test time stability varies a lot across reasoners. While FaCT++ measurements are mostly stable, Pellet measurements vary a lot across almost all potential categories. The pathological cases as described in section 4, EHC and EHH, occur rarely. Out of the 39,894 tests that got easier overall, only 5,620 are of a high magnitude. Out of those, 3,322 are clear cut, and 1,106 of high stability. From the clear cut cases, only 24 are harder than 100 ms, and only 2 are harder than a second. From the highly stable cases, 40 are harder than 100 ms and none harder than a second. Out of the 3,322 clear cut cases, only 116 are potentially not affected by non-deterministic behaviour of the algorithms (none of the high stability cases), and the hardest one is just about 22 ms.

## 7 Discussion

We quantify the impact of subsumption test hardness (RQ1) on classification time in two ways: (1) Contribution of test times measured to OCT and (2) ratio of number of ontologies with tests to those without. In Figure 4 we can see that only few of the 136 ontologies with tests were dominated by test hardness: Only 1 ontology had more than a 50% contribution of total SST for Hermit, 7 for Pellet, 19 for FaCT++ and 23 for JFact. However, there are cases where the contribution is very high. The ratio of ontologies entirely without tests is very high: FaCT++: 52%-71%, HermiT 55%-80%, JFact in 56%-76% and Pellet 64%-84%. We have established only the lower bound. The upper bound covers the very unlikely possibility that the failed classifications might be all without

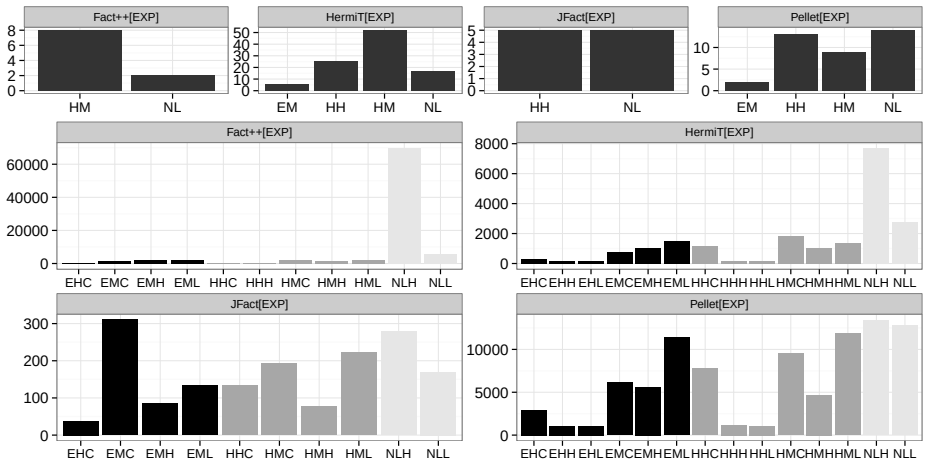


**Fig. 7.** Hardness classes by reasoner. Top row: OCT, bottom 4: SST. Bin labels x-axis: 1st letter: tendency (easier, neutral, harder), 2nd: magnitude (low, medium, high), 3rd: stability: (clearcut, high, low). Y-axis: number of comparisons.

tests. Additionally, 36 of the ontologies entirely without tests have less than 100 TBox axioms (12 less than 10).

RQ2 is quantified by ratio of positive to negative tests. Positive tests account to between 0.12% (Pellet) and 2.61% (FaCT++) of the overall number of tests (JFact 2.49%, HermiT 2.12%). This low ratio is not surprising, given that the worst case $N^2$ is dominated by far by non-subsumptions. As a side observation, current traversal algorithms appear highly efficient. Only 3 ontology-reasoner pairs (two distinct ontologies, small TBoxes) trigger more than 10% of the worst case $N^2$ number of subsumption tests, and 50 pairs (30 unique ontologies) trigger more than 1% of the worst case. This result however is only indicative of the efficiency, as we do not guarantee to measure all tests.

The distribution of test hardness as shown in Figures 2 tends towards easy tests (RQ3). Table 6.1 and Figure 2 show that the number of really hard tests are in the minority: only 346 out of 2,671,896 tests measured overall are harder than a second. This result may emphasise the importance of test avoidance over further optimising individual subsumption tests. However, as there individual tests that can make up to 25% of the overall reasoning time (Fig. 4), it cannot be disregarded. Figure 3 provides a view from the level of the ontology. Again, we have only 11 ontology-reasoner pairs (10 unique ontologies) that trigger tests that are harder than 1 second during classification. While this is not indicative of the overall impact of hardness that could be also due to a large amount of easier tests, it can provide reasoner developers with hard cases to study, but also, again, might invite shifting the emphasis on test avoidance.

The experiment discussed in the next section has not been designed as a comparative study between the reasoners. The distinction by reasoner in the

presentation of the data is purely informative, as the data was gathered mostly for different ontologies and in different quantity. The *variation of the measurements*, both for individual tests and overall times, is, at least in its magnitude, surprising (RQ4). While the variation of test times could be so high merely due to the low number of measurements that are very vulnerable to experimental error (for example an unaccounted for system background kicking in, stochasticity in the garbage collection, room temperature), we cannot claim the same for the variation in the numbers of triggered tests. That 75% of the modules in the sample vary in the number of tests is a very strong indicator for the stochasticity of the classification process (at least in this particular sample), be it due to random effects in the programming language or deliberate randomness induced by the implementation. This poses a serious threat for single-run benchmarking, as it is still general practice in the DL community. A small indication of the potential impact of a particular programming environment is the very low average variation in test times collected for FaCT++, which is the only reasoner in the set implemented in C++. In the *inter-module comparison* we learned that our pathological cases rarely happen and if so, the effect they might have on overall classification time is negligible, due to the potential degree of the effect and the rarity in which they occur. Furthermore, the strong evidence of stochasticity of the classification process makes it unclear whether the effect might not simply be due to non-determinism. Despite having detected some cases that are clearly counter-intuitive (in the sense of getting easier when irrelevant stuff is added in), we cannot be sure whether modularity is the cause, due of the small effect size (RQ6). On top of that, easier and harder tests almost balance each other out. Given our sample bias, our results are not conclusive.

## 8 Conclusions and Future Work

In this paper we have presented a procedure for reliable and reproducible isolation of counter-intuitive reasoning behaviour on subsumption tests during classification and presented some such isolated cases. Future work includes completing the full characterisation of the corpus with respect to the pathological cases and then investigating the causal basis of those cases. The most likely explanation is that the additional axioms trigger a cheaper choice in the complex non-determinism algorithms. The big challenge is whether any progress can be made in a fairly reasoner independent way. One idea is to extract the justifications for a given entailment and see whether they are disproportionally difficult individually. This would suggest that the additional information is directing the algorithm toward “easier” reasoners. Another common problem of OWL reasoner benchmarking that also might have a large effect of subsumption test hardness is correctness. We are thinking about possibilities to improve the current standard (majority voting) by using a justification based approach.

## References


1. F. Baader, B. Hollunder, B. Nebel, H.-J. Profitlich, and E. Franconi. An empirical analysis of optimization techniques for terminological representation systems. *Appl. Intell.*, 4(2):109–132, 1994.
2. K. Dentler, R. Cornet, A. t. Teije, and N. d. Keizer. Comparison of reasoners for large ontologies in the OWL 2 EL profile. *Semantic Web*, 2(2):71–87, 2011.
3. B. Glimm, I. Horrocks, and B. Motik. Optimized Description Logic Reasoning via Core Blocking. In *IJCAR 2010*, pages 457–471, 2010.
4. B. Glimm, I. Horrocks, B. Motik, R. Shearer, and G. Stoilos. A novel approach to ontology classification. *J. Web Sem.*, 14:84–101, 2012.
5. B. Glimm, I. Horrocks, B. Motik, G. Stoilos, and Z. Wang. HermiT: An OWL 2 Reasoner. *J. Autom. Reasoning*, 53(3):245–269, 2014.
6. R. S. Gonalves, S. Bail, E. Jimnez-Ruiz, N. Matentzoglu, B. Parsia, B. Glimm, and Y. Kazakov. ORE Reasoner Evaluation (ORE) Workshop 2013 Results: Short Report. In *ORE 2013*, pages 1–18, 2013.
7. R. S. Gonalves, N. Matentzoglu, B. Parsia, and U. Sattler. The Empirical Robustness of Description Logic Classification. In *ISWC 2013*, pages 277–280, 2013.
8. R. S. Gonalves, B. Parsia, and U. Sattler. Performance Heterogeneity and Approximate Reasoning in Description Logic Ontologies. In *ISWC 2012*, pages 82–98, 2012.
9. B. C. Grau, C. Halaschek-Wiener, and Y. Kazakov. History Matters: Incremental Ontology Reasoning Using Modules. In *ISWC 2007*, pages 183–196, 2007.
10. B. C. Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Modular Reuse of Ontologies: Theory and Practice. *J. Artif. Intell. Res. (JAIR)*, 31:273–318, 2008.
11. I. R. Horrocks. *Optimising tableaux decision procedures for description logics*. PhD thesis, Citeseer, 1997.
12. Y. Kazakov. RIQ and SROIQ Are Harder than SHOIQ. In *KR 2008*, pages 274–284, 2008.
13. N. Matentzoglu, B. Parsia, and U. Sattler. An Empirical Investigation of Difficulty of Subsets of Description Logic Ontologies. In *DL 2014.*, pages 659–670, 2014.
14. N. F. Noy, N. H. Shah, P. L. Whetzel, B. Dai, M. Dorf, N. Griffith, C. Jonquet, D. L. Rubin, M.-A. D. Storey, C. G. Chute, and M. A. Musen. BioPortal: ontologies and integrated data resources at the click of a mouse. *Nucleic Acids Research*, 37(Web-Server-Issue):170–173, 2009.
15. A. A. Romero, B. C. Grau, and I. Horrocks. MORe: Modular Combination of OWL Reasoners for Ontology Classification. In *ISWC 2012*, pages 1–16, 2012.
16. V. Sazonau, U. Sattler, and G. Brown. Predicting Performance of OWL Reasoners: Locally or Globally? In *KR 2014*, 2014.
17. R. Shearer and I. Horrocks. Exploiting Partial Information in Taxonomy Construction. In *ISWC 2009*, pages 569–584, 2009.
18. E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. *J. Web Sem.*, 5(2):51–53, 2007.
19. A. Steigmiller, T. Liebig, and B. Glimm. Konclude: System description. *J. Web Sem.*, 27:78–85, 2014.
20. D. Tsarkov and I. Horrocks. FaCT++ Description Logic Reasoner: System Description. In *IJCAR 2006*, pages 292–297, 2006.
21. D. Tsarkov and I. Palmisano. Chainsaw: a Metareasoner for Large Ontologies. In *ORE 2012*, 2012.