

A Workbench for Anytime Reasoning by Ontology Approximation

With a case study on instance retrieval

Gaston Tagni, Stefan Schlobach, Annette ten Teije, Frank van Harmelen, Giorgios Karafotias¹

Abstract. Reasoning is computationally expensive. This is especially true for reasoning on the Web, where data sets are very large and often described by complex terminologies. One way to reduce this complexity is through the use of approximate reasoning methods which trade one computational property (e.g. quality of answers) for others, such as time and memory. Previous research into approximation on the Semantic Web has been rather ad-hoc, and we propose a framework for systematically studying such methods. We developed a workbench which allows the structured combination of different algorithms for approximation, reasoning and measuring in one single framework. As a case-study we investigate an incremental method for instance retrieval through ontology approximation, and we use our workbench to study the computational behaviour of several approximation strategies.

1 INTRODUCTION

Motivation: Since the introduction of anytime algorithms in [1] it has become widely accepted that they are attractive for many reasoning tasks in AI [9]. Instead of producing the perfect answer after a long period of computation, they allow a reasoning task to progress gradually, producing output of increasing quality as run-time progresses. This allows to produce meaningful output under time-pressure, and to save time for applications where an approximate answer is already sufficient.

A recent set of reasoning challenges has been posed to AI by Semantic Web applications. These applications typically use large and complex ontologies for searching information on the Web, personalising Web sites, matchmaking between web-services, etc. They rely on reasoning in languages based on Description Logics [3], such as the retrieval of instances of a specified class (called instance-retrieval). Many of these Semantic Web applications are performed under time pressure (eg. because of user-interaction). Often approximate answers are sufficient, given the incomplete and noisy nature of the data on the Web. Clearly approximate anytime reasoning is an important step towards a scalable Semantic Web.

A workbench for studying ontology approximation: The need for approximation on the Semantic Web raises the challenge to develop algorithms for anytime Semantic Web reasoning, and several attempts have been made to find suitable approximation strategies and study their effects in practice [7, 2, 6, 8]. Until now, this work has been limited in scope, has had a rather ad-hoc character (lacking a general framework for theory and application), and most importantly, results have often been inconclusive and show a need for

a more thorough experimental analysis. A systematic evaluation of strategies and heuristics is challenging, and the results until now have been difficult to reproduce and compare. In this paper, we introduce a framework and a workbench for testing approximation algorithms systematically that will make the development of such methods more easy, and thus increases their chances of adoption and deployment.

To this end, we design a workflow consisting of a number of independent modules, which can be instantiated for different reasoning tasks and approximation strategies. The crucial elements of this workflow are separate modules for *approximation*, *reasoning* and *evaluation*. The first allows implementing approximation strategies based on subsetting (eg. subsets of axioms, or of vocabulary), the second allows to specify a specific reasoning task (eg. instance retrieval, ontology classification, or consistency checking), and the final module allows to implement a suitable evaluation metric. This metric results in a novel type of gain diagrams, which are a further contribution of this paper. This 3-step design is the basis for an adaptable workbench, which is implemented and made available online.

A case-study in Instance Retrieval: We evaluated our framework with a study of a particular Semantic Web reasoning task: Instance Retrieval. For this exercise we studied the behaviour of half a dozen different approximation strategies on over 30 different real-world ontologies available on the Web.

Finding instances of a concept with respect to a complex ontology is known to be a computationally difficult task. Very often users are satisfied with an algorithm that returns some, but maybe not all, answers very fast, instead of waiting for the full answer-set to be returned after a longer wait. This makes instance retrieval an interesting case-study for our anytime approximation framework.

The basic idea of our approach is to select a subset of the vocabulary of an ontology, use this set to create a new, approximate version of the ontology by rewriting the set of terminological axioms and to gradually increase the size of the vocabulary. Instances of the ontology can then be retrieved using the approximate version of the ontology. This mechanism yields an anytime algorithm for retrieving instances of an ontology that produces sound but incomplete results where completeness increases as time progresses and the size of the vocabulary increases. This paper presents the results of a series of experiments aimed at empirically evaluating a number of selection strategies in order to discover how each strategy affects the anytime retrieval of instances.

Related work: The idea of selecting a subset of the vocabulary has been formalised as approximate deduction in [5]. These results have been used in earlier work on approximate subsumption by [2] and [7], and we will apply a rewrite procedure defined in [7]. The essential difference with our approach is that in [2, 7] the approximation

¹ Department of Computer Science, Vrije Universiteit Amsterdam, The Netherlands, email: gtagni@few.vu.nl

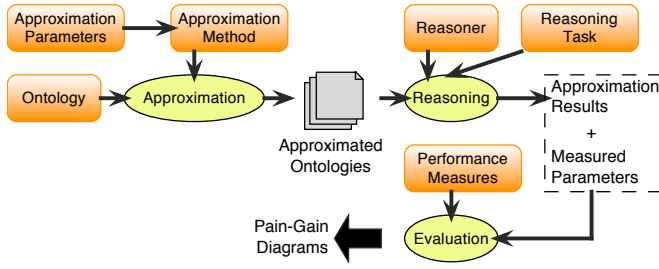


Figure 1: 3-step workflow for approximate reasoning experiments; every square box can be changed per experiment

is used to reformulate the *queries*, whereas we use it to approximate the *ontology*. The results in [2] are mostly negative, while [7] does not report any empirical results. Our experiments show that approximating the ontology produces much better anytime behaviour than the reported results on approximating the query. Related to the topic of approximate Abox reasoning is also the work by [4] where the authors propose a soundness-preserving approximate reasoning approach for OWL2 ontologies. The main difference with our work is that such approach is based on a language weakening technique.

Findings: An observation that confirmed previous, less thorough studies, is that structural syntactic approximation can work astonishingly well in a number of cases, but not in all. Although we believe to have tested the most natural selection strategies we have not found a strategy that works universally. Whether or not approximation works at all depends mostly on the ontology and seems mostly independent of the strategy used, while the actual amount of gains are strongly dependent on the particular strategy that is being used.

What to expect from this paper: This paper has two major contributions, a domain specific one, where we provide an assessment of approximation strategies for a particular Semantic Web reasoning task, and a methodological one by introducing a standardised way to perform comparative experiments for related reasoning problems in a greatly simplified fashion through our workbench.

In the following we will first describe our framework in Section 2, before describing our case-study on approximate instance retrieval in Section 3, including the necessary theory, the experimental setup and results, as well as our findings. We will analyse the pros and cons of our framework in Section 4 before we conclude in Section 5.

2 FRAMEWORK

Our framework consists of a pipeline of three steps (see Figure 1), resulting in a new type of gain diagrams. These three steps allow to define (i) the particular approximation heuristic to be used, (ii) the reasoning task to which it should be applied, and (iii) the definition of a performance measure for evaluating the heuristic. This section describes each of these steps and the resulting gain diagrams.

2.1 Approximation step

The foundational results from [5] show that performing an approximate reasoning task on a logical theory can be transformed into executing a classical reasoner on a suitably approximated theory. Hence, the purpose of the approximation step is to take an ontology O , an approximation method M and to return a sequence of approximations $O_1, O_2, \dots, O_n$ computed according to M and sorted according to some criterion. Approximation methods can take as input multiple parameters that drive the approximation method. For instance, the

approximation method used in our experiments takes as input a parameter that indicates the specific vocabulary selection strategy to be used for computing the approximations. Approximations can be sorted according to different criteria such as their completeness w.r.t the original ontology (from less complete approximations to more complete ones) or their size in terms of axioms. The framework imposes no restrictions on the criteria used for sorting approximations. The only restriction imposed in this respect is that such an ordering scheme reflects the complexity of the problem one is trying to solve. Very often, such approximations can be phrased in terms of a selection method, operating on either the symbols appearing in an ontology (vocabulary selection), or on the set of axioms in an ontology (axiom selection), operating on either or both of the A-box and T-box of the ontology. Our framework does not depend on a particular strategy: the only requirement is that for a given ontology, the approximation module returns a sequence of approximations that follow a certain ordering schema.

Although our framework imposes no further constraints on the approximation step except that it produces ontologies that can be used in the reasoning step, some formal properties of such selection steps are desirable. Let O^* denote the semantic closure of an ontology, ie. all facts that can be derived according to its semantics. We then have *soundness* if each $O_i^* \subseteq O^*$, ensuring that the approximate results are correct (although possibly incomplete); *monotonicity* if $O_i^* \subseteq O_{i+1}^*$ for all $i = 1, \dots, n-1$, ensuring that the successive approximations get more correct; and *completeness* if $O^* = O_n^*$, at which point the approximation has reached perfect quality.

2.2 Reasoning step

Approximation can be applied to different reasoning problems such as Instance retrieval or Classification. All that our framework requires is that the reasoning step takes as input an ontology, and returns *answer-sets*. These answer sets could be instance-class memberships (for instance retrieval) or class-class subsumptions (for classification). It is these answer-sets that determine the quality of the approximation, and the computational efforts which determine the price one has to pay.

2.3 Evaluation step

In the analysis step of our framework we specify the notions of success and costs. These performance measures can be eg. the standard notions of recall (the number of retrieved facts in relation to all possible findings), or precision (the correctness of the given answers), or some more non-standard notion of semantic proximity of the approximate answers to the perfect answers. More generally, we propose a notion of *gain*, which abstracts over the detailed measures and describes the results as ratios between possible and actual findings. *Pain* is the orthogonal notion describing the ration between the costs of reasoning over an approximate ontology versus the non-approximate one. For specific examples of pain one could think of costs in terms of runtime or other computational resources, such as memory, user-interaction, database access, etc.

Gain-Pain diagrams: Obviously, we are interested in whether the gain (success-ratio of current answers against perfect answers) outweighs the pain (cost-ratio of current answers against perfect answers), in other words in the gain-pain difference. This ratio is plotted in our gain-pain diagrams which show at which point of the anytime computation the gain outweighs the pain (or not, as the case may be, and by how much). Figure 2 illustrates these measures.

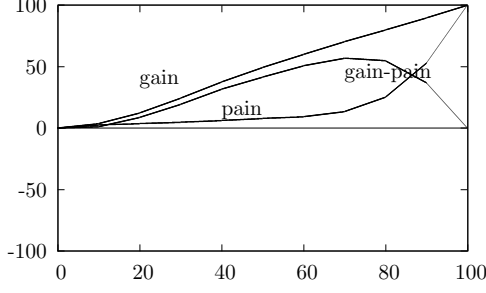


Figure 2: gain, pain and gain-pain curves

As the quality of the approximation increases along the x-axis from 0 – 100%, in this example the gain increases linearly while the pain increases much more slowly initially, and rises more sharply in the final 20%. The combined performance measure (pain-gain curve) is calculated as the difference between these two, with the best performance achieved at about 75% of the approximation where the proportional gain maximally outweighs the proportional pain.

The ideal gain-pain curve rises sharply for the initial approximations of the input representing the desired outcome of a high gain and low pain in the early stages of the algorithm. Although such a convex gain-curve is the most ideal, even a flat gain curve at $y = 0$ is already attractive, because it indicates that the gains grow proportionally with costs, giving still an attractive anytime behaviour.

Notice that gain-curves always start in $(0,0)$, since for the empty input both gain (eg. recall) and pain (eg. runtime) are 0, hence their difference is 0. Gain curves always ends in $(100,0)$, since for the final perfect approximation both recall and runtime are 100%, hence their difference is again 0. Also notice that gain-pain curves can be negative when the proportional pain outweighs the proportional gain for certain approximations.

In our experiments in section 3, we will run different approximation heuristics on an ontology, and we will report their results in terms of gain-pain curves. It is not obvious how to compare such curves. Higher curves are to be preferred over low curves, and curves with an early peak are to be preferred over curves with a late peak. In order to compare the curves of multiple strategies we associate the curve G with a *gain-score*, defined as the product of the area under the gain-curve and the point at which the absolute gain is maximal, more specifically $gainscore = area * (1 - P)$, where P is the percentage of vocabulary at which the absolute gain is maximal. In this way we reward strategies with a flat gain-curve that reach maximum gain earlier in the reasoning process (and of course, gain-scores can be negative in case the pain outweighs the gain).

3 APPROXIMATE INSTANCE RETRIEVAL

In this section we report on a series of experiments aimed at studying a specific reasoning task, namely: instance retrieval. This is done by using the framework introduced in the previous section and instantiating its three main components. In particular, the *approximation* component is instantiated with a variety of vocabulary selection strategies to produce different sound but incomplete approximations of the same ontology. The *reasoning task* component is instance retrieval, and the *evaluation* module is instantiated with comparing recall of instance retrieval against runtime.

3.1 A sound approximation for instance retrieval

In this section we will define a sound and incomplete approximation for anytime instance-retrieval. An ontology is a set of axioms. To simplify the presentation in the paper these axioms have the form $A_i \sqsubseteq B_i$, with A_i and B_i built from atomic concepts and a number of logical operators, eg. conjunction, disjunction, negation, role and number restrictions, etc. Range and domain restrictions can be translated into this form. Axioms not involving classes exist, but remain unchanged in the approximation. The semantics of an ontology is given as a set of interpretations, ie. functions with domains that satisfy all its axioms, called *models*.

Given an ontology $O = (\mathcal{T}, \mathcal{A})$, consisting of a terminological part (called TBox), and an assertional part (called ABox), the task of instance retrieval is to find all individuals i in O which belong to a particular concept C for all models of O , written as $O \models i : C$. The basic intuition of our approximation is that if we rewrite a terminology $\mathcal{T}$ to a weaker $\mathcal{T}'$. Then establishing $\mathcal{T}', A \models i : C$ is sufficient to establish $\mathcal{T}, A \models i : C$ for any individual i of an ABox A and concept C . In other words, entailment under $\mathcal{T}'$ is a sound (but incomplete) approximation of entailment under $\mathcal{T}$.

We will now define a rewrite procedure that we will apply to every axiom $A_i \sqsubseteq B_i$ of $\mathcal{T}$ in order to obtain $\mathcal{T}'$. Following the ideas [5] we define an *approximation set* S consisting of a subset of the vocabulary used in $\mathcal{T}$. In this paper we will only approximate concept names. Role and individual names will remain unchanged. The rewrite procedure will restrict the vocabulary of $\mathcal{T}$ only to atoms that appear in S .

Definition 1 ([7]) The rewrite procedures $(\cdot)^S$, $(\cdot)^{S+}$ and $(\cdot)^{S-}$ are defined as follows²: $(A \sqsubseteq B)^S = A^{S-} \sqsubseteq B^{S+}$,

$$\begin{aligned} A^{S-} &= A \text{ if } A \in S & A^{S+} &= A \text{ if } A \in S \\ A^{S-} &= \perp \text{ if } A \notin S & A^{S+} &= \top \text{ if } A \notin S \\ (\neg C)^{S-} &= \neg C^{S+} & (\neg C)^{S+} &= \neg C^{S-} \\ (C \sqcap D)^{S-} &= C^{S-} \sqcap D^{S-} & (C \sqcap D)^{S+} &= C^{S+} \sqcap D^{S+} \\ (C \sqcup D)^{S-} &= C^{S-} \sqcup D^{S-} & (C \sqcup D)^{S+} &= C^{S+} \sqcup D^{S+} \\ (\exists R.C)^{S-} &= \exists R.C^{S-} & (\exists R.C)^{S+} &= \exists R.C^{S+} \\ (\forall R.C)^{S-} &= \forall R.C^{S-} & (\forall R.C)^{S+} &= \forall R.C^{S+} \end{aligned}$$

Both $(\cdot)^{+S}$ and $(\cdot)^{-S}$ terminate as the complexity of the formula decreases in any application of the rules. The TBox $\mathcal{T}^S$ is obtained by applying this rewrite procedure to every axiom in $\mathcal{T}$.

We will use a small example to illustrate our idea.

Example 1 Let $\mathcal{T} = \{A \sqsubseteq B \sqcap C, B \sqsubseteq D, C \sqsubseteq E \sqcup D\}$. If we take as successive approximation sets $\emptyset, \{B\}, \{A, B\}, \{A, B, D\}, \{A, B, C, D\}, \{A, B, C, D, E\}$, then rewriting produces approximate TBoxes as follows:

$$\begin{aligned} \mathcal{T}^\emptyset &= \{\perp \sqsubseteq \top \sqcap \top, \perp \sqsubseteq \top, \perp \sqsubseteq \top \sqcup \top\} \\ \mathcal{T}^{\{B\}} &= \{\perp \sqsubseteq B \sqcap \top, B \sqsubseteq \top, \perp \sqsubseteq \top \sqcup \top\} \\ \mathcal{T}^{\{A, B\}} &= \{A \sqsubseteq B \sqcap \top, B \sqsubseteq \top, \perp \sqsubseteq \top \sqcup \top\} \\ \mathcal{T}^{\{A, B, C\}} &= \{A \sqsubseteq B \sqcap C, B \sqsubseteq \top, C \sqsubseteq \top \sqcup \top\} \\ \mathcal{T}^{\{A, B, C, D\}} &= \{A \sqsubseteq B \sqcap C, B \sqsubseteq D, C \sqsubseteq \top \sqcup D\} \\ \mathcal{T}^{\{A, B, C, D, E\}} &= \mathcal{T} \end{aligned}$$

This rewrite procedure is taken from [7]. The essential difference is that in [7] the procedure is used to approximate the *queries* (ie.

² Only the rewrite rules for the operators in $\mathcal{ALC}$ are reproduced here. They can easily be extended to full OWL.

$\mathcal{T} \models \phi^S$), whereas we use it to approximate the *ontology* (ie. $\mathcal{T}^S \models \phi$).

The following property is crucial to establish that $\mathcal{T}^S$ is a sound approximation of $\mathcal{T}$:

Theorem 1 (From [7]) *For any axiom $C \sqsubseteq D$: if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ then $(C^{S-})^{\mathcal{I}} \subseteq (D^{S+})^{\mathcal{I}}$ for any interpretation $(\cdot)^{\mathcal{I}}$.*

The intuition behind this is that $(C^{S-})^{\mathcal{I}} \subseteq C^{\mathcal{I}}$ (since the atoms in C not listed in S have been replaced by $\perp$), and $D^{\mathcal{I}} \subseteq (D^{S+})^{\mathcal{I}}$ (since the atoms in B_i not listed in S have been replaced by $\top$). The full proof of this is given in [6]. From this the following is immediate:

Corollary 1 (Soundness) *For any subset S of the concept-names occurring in $\mathcal{T}$, and individual names i occurring in an ABox $\mathcal{A}$: if $(\mathcal{T}^S, \mathcal{A}) \models i : C$ then $(\mathcal{T}, \mathcal{A}) \models i : C$*

If $S = \emptyset$, $\mathcal{T}^S$ is reduced to the empty (trivial) TBox, entailing only tautologies. Similarly, if S contains all atoms from $\mathcal{T}$ then the rewrite operation is the identity, and the consequences of $\mathcal{T}^S$ equal those of $\mathcal{T}$. Note that in these experiments we do not approximate the ABox. Although risking to overload notation we will use D^S to abbreviate $\{i \in \mathcal{A} \mid (\mathcal{T}^S, \mathcal{A}) \models i : D\}$, ie. the set of all individuals classified as instances of D wrt. the approximated TBox $\mathcal{T}^S$. In general, if S grows, the entailments from $\mathcal{T}^S$ become more complete:

Theorem 2 (Monotonicity) *If $S_1 \subseteq S_2$ then $(\mathcal{T}^{S_1}, \mathcal{A}) \models i : C$ entails $(\mathcal{T}^{S_2}, \mathcal{A}) \models i : C$. In other words, $C^{S_1} \subseteq C^{S_2}$ for any C .*

This is because any model for $(\mathcal{T}^{S_2}, \mathcal{A})$ is necessarily also a model for $(\mathcal{T}^{S_1}, \mathcal{A})$.

Anytime instance retrieval We can now obtain an anytime algorithm for instance-checking wrt. a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$ by starting out with finding instances wrt. $\mathcal{T}^S$ and $\mathcal{A}$ for an initial (typically small) set S . We then increase S and repeat the procedure until either the number of retrieved instances is sufficient for our purposes, or we run out of available computing time, or S contains all atoms from $\mathcal{T}$. Theorem 2 guarantees that the output of this algorithm monotonically improves during the iteration, as is typically required of anytime algorithms [9]

Example 2 Let $\mathcal{T} = \{A \sqsubseteq B \sqcap C, B \sqsubseteq D, C \sqsubseteq E \sqcup D\}$ be defined as above, and $\mathcal{A} = \{i : A, j : C \sqcap \neg E, k : B\}$. Retrieving all instances of D returns the set $\{i, j, k\}$. Taking the ordering from the previous example we get

$$D^\emptyset = D^{\{B\}} = D^{\{A, B\}} = D^{\{A, B, C\}} = \emptyset \\ D^{\{A, B, C, D\}} = \{i, k\} \quad D^{\{A, B, C, D, E\}} = \{i, j, k\}$$

This example illustrates that for small values of S , $\mathcal{T}^S$ is a very incomplete approximation of $\mathcal{T}$; with increasing S , $\mathcal{T}^S$ becomes a less incomplete approximation; and when S contains all atoms from $\mathcal{T}$, $\mathcal{T}^S$ equals $\mathcal{T}$. The instance-retrieval algorithm given above increases S in successive iterations. The choice of how to increment S determines how quickly the approximation approaches the classical result. This is shown in the following example:

Example 3 Let $\mathcal{T}$ be the same as in ex. 1, but now with the sequence $S = \emptyset, \{B\}, \{B, D\}, \{A, B, D\}, \{A, B, C, D\}, \{A, B, C, D, E\}$. This yields the following set retrieved instances for D :

$$D^\emptyset = D^{\{B\}} = \emptyset \quad D^{\{B, D\}} = \{k\} \quad D^{\{A, B, D\}} = \{i, k\} \\ D^{\{A, B, C, D\}} = \{i, k\} \quad D^{\{A, B, C, D, E\}} = \{i, j, k\}$$

3.2 Experimental setup

The example from the previous section illustrates that different approximation strategies result in different anytime behaviours of the retrieval algorithm. This raises the question on what would be a good approximation strategy. In this section, we will define different strategies and we will use the 3-step framework from section 2 to investigate their resulting anytime behaviour on a number of realistic ontologies.

3.2.1 Selection strategies

Given a vocabulary set V of atomic concept names, a selection function returns a subset $V_i \subseteq V$. In an anytime setting, the set V_i is computed from those concepts in V that were not selected previously. In our experiments we tested six selection functions, namely:

- *Random (R)*: This function randomly selects a set of atomic concept names.
- *Most Referenced (MR)*: This function selects concept names according to the number of times they appear in terminological axioms.
- *Most Members (MM)*: In each approximation step concept names are sorted according to the number of instances that were retrieved in the previous step. Initially, concepts are sorted according to the Most References strategy. The rationale behind this strategy is to select as early as possible those concepts that can produce the largest number of instances.
- *Restriction Class (RC)*: This function gives higher priority to the fillers of quantified concept expressions and to their respective sub concepts. If the number of such elements is less than desired number M the additional concepts are chosen based on the number of instances asserted in the assertional part of the ontology. The rationale of this strategy is that property restrictions are used for defining classes implicitly. Consequently, these classes may contribute to retrieving a large number of instances. The main disadvantage of this strategy is that not every class in an ontology is defined through property restrictions, a characteristic that makes this strategy incomplete. Therefore, as with the previous strategy this one needs to be complemented with another strategy for selecting classes that are not defined through property restrictions.
- *Most Direct Subclasses (MDS)*: This function selects atomic concepts based on the number of direct subclasses they have. The first time this strategy is used, atomic concepts are sorted in decreasing number of direct subclasses and each successive call to this function returns the next set of concepts. As with the Most Referenced strategy concepts can be sorted only once at the beginning of the anytime reasoning process.
- *Least Direct Subclasses (LDS)*: This function is the opposite of the MDS function. The rationale for this strategy is that concepts with the least number of subclasses are more specific and tend to be used to annotate large number of individuals.

3.2.2 Reasoning step

Most of our tests were performed using Pellet (v2.0.0rc7)³, a well-known open source DL reasoner capable of dealing with OWL 2 ontologies. Although Pellet performed well with many of the ontologies tested it showed some problems when reasoning with some of them, eg. LKIF, for which the reasoner was unable to classify the ontology

³ <http://clarkparsia.com/pellet/>

even after a long period of time. To overcome these issues we used FaCT++ (v1.3.0)⁴, another standard DL reasoner. All the tests were performed on a dual-core AMD Opteron processor at 2.8Ghz. with 32Gb of RAM running Linux and Java runtime 1.6 Update 12.

Given an ontology $O = (T, A)$ with Tbox T and Abox A and a selection strategy F we generated a number of approximations T_i^S of T , each of them based on a vocabulary set S with a fixed 10% step-size. For each pair (T_i, A) we ran the reasoner ten times and measured the average value of several parameters such as recall, classification time, approximation time and instance retrieval time, among others. All experiments were run as “contract algorithms” [9], that is: creating fresh runs for each iteration, not incrementing from the previous iteration (which would constitute an “interruptable algorithm”).

3.2.3 Performance measures

In order to evaluate how each strategy influences the instance retrieval process we need to define a performance measure. Notice that since the approximation algorithm is sound we only need to measure completeness. Measuring the degree of completeness is equivalent to measuring recall, ie. the ratio between the number of instances retrieved and the total number of instances in the ontology:

$$Recall = \frac{\#InstancesRetrieved}{\#Instances}$$

Cost is measured in terms of the runtime. This time includes classification, instance retrieval and approximation time.

3.2.4 Datasets

In our experiments we used an extensive collection of well-known ontologies among which we included: LKIF, an ontology of basic legal concepts; BVA, a basic vertebrate anatomy ontology; the Family Health History Ontology (FHHO), which facilitates the representation of health histories of persons related by biological and/or social family relationships; the Gene Regulation ontology, a conceptual model for the domain of gene regulation; the Brenda ontology, a structured controlled vocabulary for the source of an enzyme; Galen, an ontology of medical concepts; and myGrid, an ontology describing the bioinformatics research domain. The complete list of over 30 ontologies together with several related information can be found on our website⁵.

Table 1 summarises some properties of these ontologies: the number of axioms, their expressivity and the number of occurrences of operators. This table shows that we have chosen a dataset of realistic ontologies of different size (ranging from hundreds of axioms to tens of thousands of axioms), and of different logical expressivity.

3.3 Experimental results

In this section, we will investigate (1) in which cases anytime instance retrieval is effective, and (2) which of the selection strategies is most effective.

Table 2 summarises the results of our experiments: for each ontology it indicates how attractive the anytime behaviour is under the various strategies. Each column shows the *gain-score* of each strategy, calculated as defined in Section 2. Unfortunately, due to space

Table 1: Some properties of the ontologies used in our experiments

	DL expressivity	#Axioms	# $\forall$	# $\exists$	# $\sqcap$	# $\sqcup$	# $\neg$
Galen-100	SHOIQ(D)	874	3734	5606	1951	784	0
BVA	SHIF	1485	0	5	16	2	1
FHHO	ALCHIF(D)	2216	0	0	72	0	0
GRO	ALCHIQ(D)	2433	236	37	67	16	1
LKIF	SHIN	1014	47	97	23	7	2
myGrid-v1	SHOIN	2204	0	289	274	12	1
Brenda	ALE	54212	0	948	0	0	0

restrictions we can not show the results of all the experiments. For the complete set of results including the gain-pain curves for all the experiments the interested reader is referred to the website of our framework. From these results we can observe the following:

Table 2: Summary of success and failure of the different strategies.

ontology	R	MM	LDS	MDS	RC	MR
Galen-100	-16.86	-15.69	-50.69	-4.62	-8.88	-10.67
BVA	-12.22	-18.89	-40.26	-0.40	-18.21	-16.44
FHHO	-3.65	-18.18	-44.35	-2.19	-15.40	-15.46
GRO-250	1.62	6.97	0.80	6.63	3.99	7.29
LKIF	3.95	2.53	3.52	6.69	5.95	2.94
myGrid-v1	1.67	6.98	3.04	10.90	27.49	6.57
Brenda	-0.18	-5.78	-17.23	7.72	12.19	4.92

Anytime instance retrieval benefits some cases: Figure 3 illustrates the case of the GRO ontology that exhibits very attractive anytime behaviour. The figure shows that for this complex ontology all strategies have an almost always-positive gain-curve. Other ontologies with attractive anytime behaviour are myGrid-v1, where all strategies have positive gains everywhere and LKIF (also Figure 3), where all strategies except MM and MR have positive gains; the last two strategies have positive gains up to the 80% and 90% point respectively.

Anytime instance retrieval benefit some but not all cases: Some ontologies allow for very high gain-scores under many strategies, but we also see cases with less attractive anytime behaviour. For example, on some versions of Galen and FHHO, almost all strategies have negative gains almost everywhere (see Figure 3). This means that more (percentage of) time is spent than (percentage of) correct answers are found. The same behaviour is observed in BVA where almost all strategies have negative gain. The results show that whether anytime instance retrieval is beneficial or not depends very much on the selection strategy used and the complexity and structure of the ontology at hand.

Which strategy performs best? The data is inconclusive on the question which strategy performs best. While some of the strategies perform well on some of the ontologies, their performance is rather poor on others; the only exception being LDS which was outperformed by every other strategy in 14 out of 20 ontologies and was the best strategy in only one ontology. As an example take the case of the RC strategy which strongly dominates all the others when applied to myGrid-v1 but is outperformed by MM, MDS and MR when applied to GRO-250 (see Table 2). Another example that illustrates this behaviour is the MR strategy that outperforms all others on GRO-250 but it is outperformed by almost every other strategy on myGrid-v1. An interesting case is the behaviour of the MM strategy which, given how the strategy works, should perform well in most of the ontologies but that is not the case. The reason for this is that the strategy

⁴ <http://owl.man.ac.uk/factplusplus/>

⁵ <http://www.few.vu.nl/~gtagni/aboxreasoning>

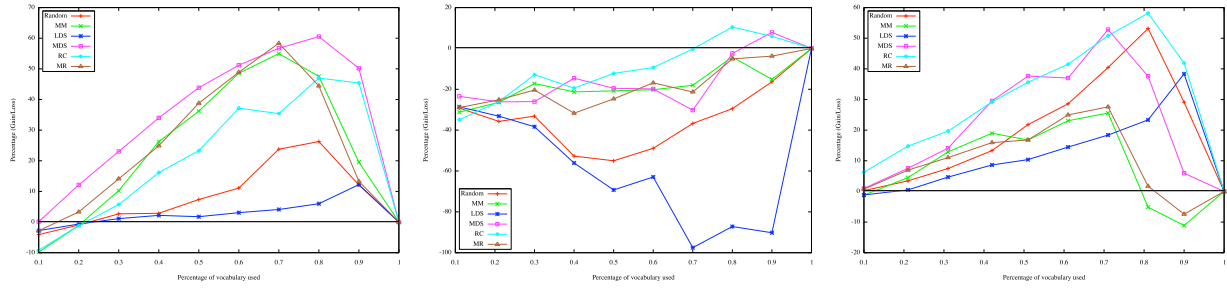


Figure 3: Gain-curves for the ontologies GRO-250, Galen-100 and LKIF-100

has a much bigger impact on the runtime (due to the sorting process) than the impact it has on the recall, and thus it reduces the gain significantly. What these experiments have shown us is that ultimately the choice of a selection strategy depends on the complexity and structure of the ontology at hand. This opens the door to future research targeted to finding better selection strategies that are able to exploit regularities in the environment (ontologies), ie. strategies or heuristics that are *ecologically valid*.

4 ANALYSING THE FRAMEWORK

Our case-study on Instance Retrieval showed the benefit of our framework for systematically analysing approximation methods. The following observations worth mentioning.

- *Reproducibility*: previous work on approximation is considerable, but the impact of published results has often been hampered by the difficulty to reproduce results, as implementing an approximation method is non-trivial, and the tuning even more so. Our framework and implementation should simplify the reuse of published results.
- *Scalability of experiments*: previously, running experiments of the kind presented in the previous section were painful experiences: combining different strategies with various reasoners on numerable ontology formats often results in incompatibilities. Experiments at the scale of our case-study, in numbers of ontologies and strategies, are made much easier in our workbench.
- *Tunability*: earlier results have shown that experiments involving reasoning with ontologies and approximation are difficult in nature. Small differences in strategy can have large impact on the results. This makes it paramount to systematically study the influence of often very subtle choices in strategy. In our framework this can now be done systematically.
- *Comparability*: often, performance measures are difficult to compare. The pain/gain diagrams introduced in Section 2 allow comparison of different approximation strategies across application problem and domain, approximation strategy and reasoning task.

5 CONCLUSION

Reasoning with complex ontologies on the Semantic Web is computationally difficult, and approximate anytime reasoning is required in many applications. In previous research it had become apparent that developing approximate methods is very difficult, there is no unique strategy to suit all ontologies, and it is not even clear whether for all ontologies approximation strategies exist.

This emphasises the need for the more systematic approach for studying approximated ontology reasoning which we provide in this paper. To this end we describe a systematic framework for studying approximate methods, and present an implementation that is publicly

available on our website. A further contribution is to develop approximation and domain independent performance measures based on so-called pain/gain diagrams, which summarise the quality of an approximation in a natural way.

We have used our systematic evaluation approach and its implementation to perform a study on instance retrieval as a prototypical case where approximation is paramount. The findings confirm previous results: none of the more structural approximations introduced in this paper work on all ontologies, and even stronger, for many ontologies all our strategies fail.

This implies that future research needs an even more fine-grained experimental set-ups, which is now simplified by our evaluation framework. We will use the workbench in future research for studying more task-driven or *ecologically valid* heuristics, ie. strategies that are based on an extensive knowledge about the ontology itself and the task for which it is being used. Doing this has become a realistic perspective due to the framework presented in this paper.

REFERENCES

- [1] Mark S. Boddy and Thomas Dean, ‘Solving time-dependent planning problems’, in *IJCAI*, pp. 979–984, (1989).
- [2] Perry Groot, Heiner Stuckenschmidt, and Holger Wache, ‘Approximating description logic classification for semantic web reasoning.’, in *ESWC*, eds., Asunción Gómez-Pérez and Jérôme Euzenat, volume 3532 of *Lecture Notes in Computer Science*, pp. 318–332. Springer, (2005).
- [3] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen, ‘From SHIQ and RDF to OWL: The making of a web ontology language’, *Journal of Web Semantics*, **1**(1), 7–26, (2003).
- [4] Yuan Ren, Jeff Pan, and Yuting Zha, ‘Towards soundness preserving approximation for abox reasoning of owl2’, in *Proceedings of the International Description Logic Workshop (DL2010)*, (2010).
- [5] Marco Schaerf and Marco Cadoli, ‘Tractable reasoning via approximation’, *Artificial Intelligence*, **74**(2), 249–310, (1995).
- [6] S. Schlobach, E. Blaauw, M. El Kebir, A. ten Teije, F. van Harmelen, S. Bortoli, M. Hobbelman, K. Millian, Y. Ren, S. Stam, P. Thomassen, R. van het Schip, and W. van Willigem, ‘Anytime classification by ontology approximation’, in *Proceedings of the workshop on new forms of reasoning for the Semantic Web: scalable, tolerant and dynamic*, ed., Ruzica Piskac et al., pp. 60–74, (2007).
- [7] Heiner Stuckenschmidt, ‘Partial matchmaking using approximate subsumption’, in *Proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI-07)*, (2007).
- [8] H Wache, P Groot, and H Stuckenschmidt, ‘Scalable instance retrieval for the semantic web by approximation’, *Lecture notes in computer science*, **3807**, 245, (2005).
- [9] Shlomo Zilberstein, ‘Using anytime algorithms in intelligent systems’, *AI Magazine*, **17**(3), 73–83, (1996).